2005

Learning discrete Hidden Markov Models from state distribution vectors

Luis G. Moscovich
Louisiana State University and Agricultural and Mechanical College, lmoscov@lsu.edu

LEARNING DISCRETE HIDDEN MARKOV MODELS FROM STATE DISTRIBUTION VECTORS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Luis G. Moscovich
B.S., Louisiana State University, 1998
May 2005

In Loving Memory of

Sara Paulina Levkov de Moscovich

and

Cecilia Tévelez de Moscovich

Acknowledgments

I first and foremost would like to express my profound gratitude to my advisor, Dr. Jianhua Chen, for her invaluable encouragement, insightful feedback, and infinite patience. It is her knowledgeable guidance and support that made this work possible. I am grateful for her assistance and constructive criticism throughout all the stages of this research.

I also would like to thank the members of my advisory committee, Dr. Daniel C. Cohen, Dr. Donald H. Kraft, Dr. Sukhamay Kundu, and Dr. John M. Tyler for their individual contributions to the overall success of my studies in general and to this work in particular; and to Dr. Ahmed A. El-Amawy for serving as the Graduate Dean's representative.

Thanks are due to the Department of Computer Science, and the Graduate School at Louisiana State University for the financial support received in the form of assistantships and fellowships that allowed me to concentrate my time and focus on the development of this research.

None of this effort would have been possible without the inspiration and constant encouragement of my wife Marina, and my father Ricardo to whom I am eternally indebted. Finally, I would like to thank —and apologize to— my daughter, Tamara, who having both parents simultaneously engaged in their respective doctoral programs, had to endure many hours of day-care since a very early age, and spend more time in front of a television set than it is humanly bearable or advisable.

Table Of Contents

List of Tables

List of Figures

List of Definitions

Abstract

Hidden Markov Models (HMMs) are probabilistic models that have been widely applied to a number of fields since their inception in the late 1960's. Computational Biology, Image Processing, and Signal Processing, are but a few of the application areas of HMMs.

In this dissertation, we develop several new efficient learning algorithms for learning HMM parameters.

First, we propose a new polynomial-time algorithm for supervised learning of the parameters of a first order HMM from a state probability distribution (SD) oracle.

The SD oracle provides the learner with the state distribution vector corresponding to a query string. We prove the correctness of the algorithm and establish the conditions under which it is guaranteed to construct a model that exactly matches the oracle's target HMM. We also conduct a simulation experiment to test the viability of the algorithm. Furthermore, the SD oracle is proven to be necessary for polynomial-time learning in the sense that the consistency problem for HMMs, where a training set of state distribution vectors such as those provided by the SD oracle is used but without the ability to query on arbitrary strings, is NP-complete.

Next, we define helpful distributions on an instance set of strings for which polynomial-time HMM learning from state distribution vectors is feasible in the absence of an SD oracle and propose a new PAC-learning algorithm under helpful distribution for HMM parameters. The PAC-learning algorithm ensures with high probability that HMM parameters can be learned from training examples without asking queries.

Furthermore, we propose a hybrid learning algorithm for approximating HMM parameters from a dataset composed of strings and their corresponding state distribution vectors, and provide supporting experimental data, which indicates our hybrid algorithm produces more accurate approximations than the existing method.

Chapter 1:
Introduction

1.1 Preliminaries

Probabilistic models are widely employed to emulate and predict the behavior of complex stochastic systems across a large number of fields. Hidden Markov Models (Rabiner 1989), in particular, have been successfully applied in such areas as Speech Processing (Rabiner 1989, Juang and Rabiner 1991) and Computational Biology (Eddy 1996, Baldi, Chauvin, Hunkapiller and McClure 1993), due to the fact that they are especially suited to represent time-varying signals of flexible length, such as speech, as well as randomized sequences, such as DNA chains. Other areas of application of HMM include Information Extraction (Scheffer, Decomain and Wrobel 2001) and Character Recognition (Vlontzos and Kung 1992). Since constructing the right model is crucial for the tasks of emulation and prediction, accurately learning HMM parameters becomes a matter of both theoretical and practical relevance.

Several approaches have been proposed for training HMMs from observations. The most widely used method for HMM parameter estimation is by means of the well-known Baum-Welch algorithm (Baum, Petrie, Soules and Weiss 1971, Baum 1972). The Baum-Welch algorithm is a dynamic programming algorithm of the Expectation-Maximization type (Dempster, Laird and Rubin 1977) for HMMs. The algorithm performs a reestimation of the HMM parameters from an initial guess in order to (locally) maximize the likelihood of a given sequence in the model. Each iteration of the algorithm converges monotonically towards local maxima.

Although initially limited to training HMM parameters from a single observation sequence, the method have since been extended to training from multiple observations. The first improvement in that direction imposed the assumption that the multiple sequences

be statistically independent (Levinson, Rabiner and Sondhi 1983). One such approach involves using the Baum-Welch algorithm separately on each individual observation to obtain several HMM estimations (one per observation) that are later combined into a single HMM (Davis, Lovell and Caelli 2002). Further combinatorial refinements to the Baum-Welch algorithm have since allowed to prescind from the independence assumption when training from multiple observations (Li, Parizeau and Plamondon 2000).

Many variations and alternative algorithms to Baum-Welch have been proposed for training HMMs that maximize the likelihood of a set of sequences. Among the most prominent of these methods are the segmental K-means algorithm for HMM (Juang and Rabiner 1990), HMM induction by Bayesian model merging (Stolcke and Omohundro 1993), gradient descent optimizations for HMM estimation (Baldi and Chauvin 1994), and class-specific Baum-Welch (Baggenstoss 2001).

The main drawback of the Baum-Welch based methods lies in a strong incidence of the choice of initial guess. Depending on the initial parameters utilized, Baum-Welch may converge to sub-optimal local maxima. Several try runs involving different initial guesses are usually required to arrive to an optimal solution.

In this work, we adopt a different approach to HMM training. All the aforementioned methods are maximum likelihood methods for HMMs. Their aim is to find optimal HMM parameters that maximize the probabilities of a dataset of observations in the model. In contrast, our approach to training HMMs involves learning HMM parameters that associate to each sequence in the set a specific probability value.

Given a set of strings, each with an associated desired probability in the model, our proposed method attempts to construct an HMM in which the probability of each string in the training dataset evaluates to the target probability value. This provides a method to construct a model that not only fits the occurrence of high probability sequences but it also accounts for the incidence of low probability strings in the training set.

In 1992, W.G. Tzeng (Tzeng 1992), proposed a supervised learning algorithm for learning the parameters of a Probabilistic Automaton (PA) using an SD oracle. In the supervised (or active or guided) learning framework (Angluin 1988), an oracle (the teacher) correctly answers the queries posed by a learning algorithm (student). Based on Tzeng's work, we propose a new efficient algorithm that, using the SD oracle, learns the parameters of an HMM. The oracle provides the learning algorithm with string probabilities in the form of state distribution vectors (string probabilities distributed over HMM states). From those state distribution vectors, the learning algorithm computes the parameters of the target model.

We show that the SD oracle is necessary for learning HMM parameters from state distribution vectors by proving a theorem stating that the consistency problem for HMM using state distribution vectors is NP-Complete. This result demonstrates that the SD oracle ability to supply the state distribution vectors of arbitrary strings is necessary for exactly learning HMM parameters from state distribution vectors. In other words, the problem of exactly learning HMM parameters from a set of state distribution vectors without such ability is intractable (under the assumption that $\mathcal{P} \neq \mathcal{NP}$). We establish a sufficient set of conditions on the target HMM under which our learning algorithm is guaranteed to find an exact solution for the target HMM.

We also define a family of helpful distributions and provide an alternative learning framework for our HMM learning algorithm under which polynomial-time learning from state distributions vectors in the absence of the SD oracle becomes feasible.

We propose a new PAC-algorithm under these helpful distributions for learning the parameters or a target HMM from a set of strings and their state distribution vectors.

In the remainder of this chapter the necessary notation and definitions will be introduced.

Chapter 2 describes the SD oracle, and presents the active learning algorithm in detail. It incorporates proof of the algorithm correctness and analysis of its complexity. Additionally, it presents the results of our simulation experiments that confirm the algorithm's viability.

In Chapter 3, the use of the SD oracle for efficient active learning will be justified based on the fact that the consistency problem for HMM, using a training dataset consisting of the same information carried by the SD oracle —state distribution vectors— is NP-Complete.

Chapter 4, elaborates alternative learning frameworks for the learning algorithm from state distribution vectors in the absence of the SD oracle and introduces our PAC-learning algorithm under helpful distributions.

In Chapter 5, we present a hybrid algorithm to approximate the parameters of an HMM from state distribution vectors that improves on a current approach for training HMM parameters from generating probabilities.

1.1.1 Notation

Let $\vec{w}[i]$ represent the i^{th} element of an arbitrary n-dimensional row vector $\vec{w}$.

Let $\vec{W}[i]$ represent the i^{th} row of a matrix $\vec{W}$.

Let $\vec{W}[i,j]$ represent the element in the i^{th} row and j^{th} column of a matrix $\vec{W}$.

Given two arbitrary $(n \times m)$-dimensional matrices $\vec{V}$ and $\vec{W}$, it will be written $\vec{V} \geq \vec{W}$ to denote that $\forall(\, 1 \leq i \leq n,\ 1 \leq j \leq m) :\ \vec{V}[i,j] \geq \vec{W}[i,j]\ .$

Let $\vec{0}_n$ be the n-dimensional *zero row vector*: $\forall(\, 1 \leq i \leq n) :\ \vec{0}_n[i] = 0\ .$

Let $\vec{1}_n$ be the n-dimensional *one row vector*: $\forall(\, 1 \leq i \leq n) :\ \vec{1}_n[i] = 1\ .$

Let $\vec{0}_{n\times n}$ denote the $(n \times n)$ *zero matrix*: $\forall(\, 1 \leq i,j \leq n) :\ \vec{0}_{n\times n}[i,j] = 0\ .$

Let $\vec{I}_n$ denote the $(n \times n)$ *identity matrix*.

Let $\vec{v}^{\,T}$ denote the *transpose* of a vector $\vec{v}$.

Let $[\![\vec{u}, \vec{v}, \vec{w}]\!]$ denote the row vector obtained by concatenating row vectors $\vec{u}$, $\vec{v}$, and $\vec{w}$. Let $\vec{e_i}$ be an n-dimensional row vector such that:

$$\vec{e_i}[j] = \begin{cases} 1 & \text{if } j = i \ , \\ 0 & \text{if } j \neq i \ , \end{cases} \qquad \forall (\ 1 \leq i, j \leq n) \ .$$

Note that $[\vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T] = \vec{I}_n$.

1.1.2 Definitions

A (row) vector $\vec{v} = \{v_1, \ldots, v_n\}$ is *stochastic* if $\sum_{i=1}^{n} \vec{v}[i] = 1$, and $\forall (\ 1 \leq i \leq n) : \ \vec{v}[i] \geq 0$. A matrix is *stochastic* if all its rows are stochastic.

Definition 1.1. A *Deterministic Finite Automaton* (DFA) $A = (Q, \Sigma, \delta, q_1, F)$ is a 5-tuple where:

- $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,
- $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is a finite, non-empty alphabet of (input) symbols,
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function,
- $q_1 \in Q$ is the initial state,
- $F \subseteq Q$ is a set of final states.

Definition 1.2. A *Probabilistic Automaton* (PA) $R = (Q, \Sigma, \delta, \rho, F)$ is a 5-tuple where:

- $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,
- $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is a finite, non-empty alphabet of (input) symbols,
- $\delta : Q \times Q \times \Sigma \rightarrow [0, 1]$ is a transition probability function such that:

$$\sum_{j=1}^{n} \delta(q_i, q_j, \sigma) = 1 \qquad \forall (\ \sigma \in \Sigma, \ 1 \leq i \leq n) \ ,$$

– $\rho : Q \to [0, 1]$ is an initial state probability distribution such that:

$$\sum_{i=1}^{n} \rho(q_i) = 1 \ ,$$

– $F \subseteq Q$ is a set of final states.

1.2 Hidden Markov Models

A Discrete Hidden Markov Model is a symbol-generating automaton composed of a finite set of states, each of which has associated an independent probability distribution called the *Display Probability Distribution* (DPD). A starting state is chosen according to an *Initial State Probability Distribution* (ISPD). Each time a state is visited it 'emits' a symbol —an observation— from a finite alphabet according to the state's DPD. Transitions among the states follow a set of probabilities called the *Transition Probability Distribution* (TPD). The HMM output is the string of display symbols generated during this process. The states visited in emitting the strings are however not visible, and account for the 'hidden' adjective in the model's name. Unlike PA where transitions are driven by an input string of symbols, HMMs are sequence (string) generating automata. The symbol generating process is detailed in Sec. 1.2.3.

Hidden Markov Models are currently implemented as the main modeling method in such diverse and relevant applications as speech recognition (Lee, Hon, Hwang and Huang 1990), DNA profiling (Haussler, Krogh and Mian 1994, Hughey and Krogh 1996), protein modeling (Karplus, Sjolander and Sanders 1997, Krogh, Brown, Mian, Sjolander and Haussler 1993), visual recognition (Starner and Pentland 1995), and traffic surveillance (L. Eikvil 2001).

Figure 1.1 shows an example HMM with three states. A formal definition follows.

Definition 1.3. A *Hidden Markov Model* (HMM) U is a 5-tuple $U = (Q, \Sigma, \delta, \beta, \rho)$:

– $Q = \{q_1, \ldots, q_n\}$ is a finite set of states,

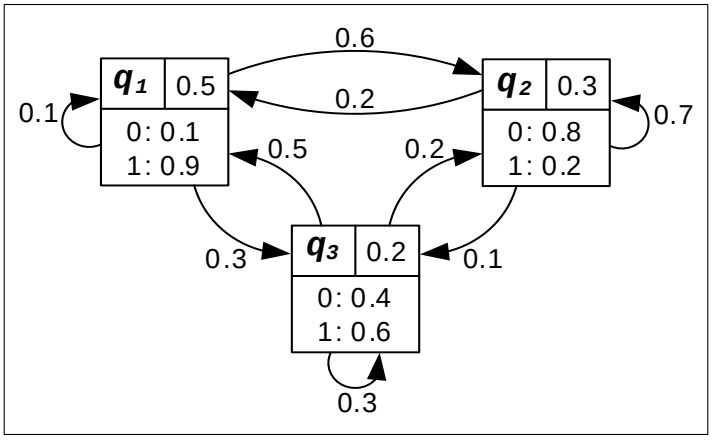

Figure 1.1: An example HMM with three states ($Q = \{q_1, q_2, q_3\}$), represented by the rectangular boxes, emitting two display symbols ($\Sigma = \{0, 1\}$). The arrows represent the TPD. The lower part of the boxes shows the DPD on each state. The top right corner of each state shows the ISPD of the corresponding state.

– $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ is a finite, non-empty alphabet of (display) symbols,

– $\delta : Q \times Q \to [0, 1]$ is a transition probability function —the TPD, such that:

$$\sum_{j=1}^{n} \delta(q_i, q_j) \ = 1 \qquad \forall (\, 1 \leq i \leq n) \ ,$$

– $\beta : Q \times \Sigma \to [0, 1]$ is a display probability function —the DPD, such that:

$$\sum_{h=1}^{m} \beta(q_i, \sigma_h) \ = 1 \qquad \forall (\, 1 \leq i \leq n) \ ,$$

– $\rho : Q \to [0, 1]$ is an initial state probability distribution —the ISPD, such that:

$$\sum_{i=1}^{n} \rho(q_i) \ = \ 1 \ .$$

Without loss of generality, the alphabet Σ will be assumed to be $\Sigma = \{0, 1\}$ unless otherwise noted. In the context of this work, the term HMM is used as a synonym

for Discrete Hidden Markov Models which are the sole focus of this research. There is, however, actual and useful distinction in the literature for HMMs where the observations at each state are allowed to follow a continuous, rather than a discrete, distribution.

1.2.1 Theory Assumptions

Several assumptions are significant to HMM Theory:

- *Markov Assumption*: The probability of the next state to be visited depends only on the current state. In other words, there is a lack of memory in the model of any previously visited states other than the current state[1].
- *Output Independence Assumption*: The symbol to be emitted by the current HMM state is statistically independent of the symbols previously displayed.
- *Stationary Assumption*: The transition probabilities are independent of the time at which the transition actually takes place.

1.2.2 Matrix Notation

For algebraic convenience, the probability distributions of an HMM U consisting of $n = |Q|$ states, and an alphabet of $m = |\Sigma|$ symbols, will frequently be expressed in matrix form (see Table 1.1) $U = (Q, \Sigma, \vec{M}, \left\{ \vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m} \right\}, \vec{p})$ where:

- The ISPD ρ is represented by the n-dimensional stochastic vector $\vec{p}$, such that:

$$\vec{p}[i] = \rho(q_i) \qquad \forall(\, 1 \leq i \leq n) \ .$$

- The TPD δ is represented by the $(n \times n)$-dimensional stochastic matrix $\vec{M}$, such that:

$$\vec{M}[i, j] = \delta(q_i, q_j) \qquad \forall(\, 1 \leq i, j \leq n) \ .$$

[1] This assumption actually transforms the HMM into a first-order HMM which is the focus of this work.

Table 1.1: Matrix notation ISPD, TPD, and DPD for the alphabet symbols 0 and 1 corresponding to the example HMM shown in Fig. 1.1.

ISPD	**TPD**	**DPD**	
$\vec{p}$	$\vec{M}$	$\vec{D}_0$	$\vec{D}_1$
$\begin{bmatrix} .5 & .3 & .2 \end{bmatrix}$	$\begin{bmatrix} .1 & .6 & .3 \\ .2 & .7 & .1 \\ .5 & .2 & .3 \end{bmatrix}$	$\begin{bmatrix} .1 & 0 & 0 \\ 0 & .8 & 0 \\ 0 & 0 & .4 \end{bmatrix}$	$\begin{bmatrix} .9 & 0 & 0 \\ 0 & .2 & 0 \\ 0 & 0 & .6 \end{bmatrix}$

– The DPD β is represented by a family of m diagonal $(n \times n)$-matrices $\left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}$, such that:

$$\vec{D}_\sigma[i,j] = \begin{cases} \beta(q_i, \sigma) & \text{if } j = i \ , \\ 0 & \text{if } j \neq i \ , \end{cases} \qquad \forall(\, \sigma \in \Sigma,\ 1 \leq i, j \leq n) \ .$$

It is important to remark that due to the stochastic nature of the functions ρ, δ, and β as described in Definition 1.3, the following equations hold:

$$\sum_{j=1}^{n} \vec{M}[i,j] = 1 \qquad \forall(\, 1 \leq i \leq n) \ , \tag{1.1a}$$

$$\sum_{\sigma \in \Sigma} \vec{D}_\sigma = \vec{I}_n \ , \tag{1.1b}$$

$$\sum_{i=1}^{n} \vec{p}\,[i] = 1 \ . \tag{1.1c}$$

1.2.3 String Generation

Let $U = (Q, \Sigma, \delta, \beta, \rho)$ be an HMM. Let s_t denote the state of U visited at time t. The process of generating a string of symbols by an HMM consists of the following steps:

– At time $t = 1$, a state from Q is chosen as the starting state according to the ISPD:

$$Pr(s_1 = q_i \mid U) = \rho(q_i) = \vec{p}\,[i] \ .$$

– Each time a state $q_i \in Q$ is visited, it emits a display symbol $\sigma_j \in \Sigma$ according to its DPD:

$$Pr(\,\sigma_j \mid s_t = q_i, U) = \beta(q_i, \sigma_j) = \vec{D}_{\sigma_j}[i,i] \ .$$

– At time $t+1$ a transition to a state $s_{t+1} \in Q$ occurs following the model's TPD:

$$Pr(s_{t+1} = q_j \mid s_t = q_i, U) = \delta(q_i, q_j) = \vec{M}[i,j] \ .$$

1.2.4 String Generating Probability

Let $x : o_1 o_2 \cdots o_k, x \in \Sigma^+$ represent a length k string of symbols from the HMM alphabet Σ. A problem of interest to HMM theory is computing the generating probability of a string, $Pr(x \mid U)$, in the model.

Let Q_k denote the set of all possible sequences of states from the state set Q of length k. Assuming all state transitions are possible, the generating probability $Pr(x \mid U)$ can be computed as:

$$Pr(x \mid U) = \sum_{\{S:S \in Q_k\}} Pr(x \mid S, U) \times Pr(S \mid U) \tag{1.2}$$

This computation however is in practice unfeasible requiring operations in the order of $2k \times |\,Q|^k$. Let $S \in Q_k, S : s_1 s_2 \cdots s_k$ denote a k-length sequence of states:

– Computing $Pr(x \mid S, U) = \beta(\,s_1, o_1) \times \beta(\,s_2, o_2) \times \cdots \times \beta(\,s_k, o_k)$ requires $|k| - 1$ multiplications.

– Computing $Pr(S \mid U) = \rho(s_1) \times \delta(s_1, s_2) \times \cdots \times \delta(s_{k-1}, s_k)$ takes $|k| - 1$ multiplications.

Therefore, computing the product $Pr(x \mid S, U) \times Pr(S \mid U)$ for each possible k-length sequences of states S takes $(|\,k| - 1) + (|\,k| - 1) + 1 = 2|\,k| - 1$ multiplications.

Since at each time $t = 1, \ldots, k$ there are $|\,Q|$ possible states to transition to, there is a total of $|\,Q|^k$ possible state sequences to generate the string x. Hence $(|\,Q|^k - 1)$ sums and $(|\,Q|^k) \times (2|\,k| - 1)$ multiplications are necessary in order to compute $Pr(x \mid U)$ from (1.2). An alternative efficient method to compute $Pr(x \mid U)$ is by means of the Forward Algorithm described in the following section.

The Forward Algorithm

The *Forward Algorithm* (Rabiner 1989) is a dynamic programming algorithm to compute the generating probability in the model of a given string. In order to obtain the generating probability of a string x, the algorithm recursively computes the state distribution vector of every prefix substrings of x. The definition of state distribution vectors —also known as forward vectors in the literature— as well as the algorithm description follows.

Definition 1.4. Given an HMM $U = (Q, \Sigma, \delta, \beta, \rho)$ and a display string $x = o_1 o_2 \cdots o_k$, $x \in \Sigma^+$, the *State Distribution Vector* $\vec{P}_U(x)$ induced by x, is the n-dimensional row vector whose i^{th} component $\vec{P}_U(x)[i]$, contains the joint probability of the string x being generated by the model, and q_i being the last state visited by the HMM in generating x —i.e o_k, the last symbol in x, is emitted by the state q_i :

$$\vec{P}_U(x)[i] = \vec{P}_U(\, o_1 o_2 \cdots o_k)[i] = Pr(\, o_1 o_2 \cdots o_k \,,\, s_k = q_i \mid U) \tag{1.3}$$

The generating probability $Pr(x \mid U)$ of string x given the model can therefore be computed as:

$$\Pr(x \mid U) = \sum_{i=1}^{n} Pr(x \,,\, s_k = q_i \mid U) = \sum_{i=1}^{n} \vec{P}_U(x)[i] \ .$$

The Forward Algorithm performs the following recursive computation:

1. For $i = 1, \ldots, n$:
$$\vec{P}_U(\, o_1)[i] = \rho(q_i) \times \beta(q_i, o_1) \ .$$
2. For $j = 1, \ldots, n$:
$$\vec{P}_U(\, o_1 \cdots o_{k-1} o_k)[j] = \sum_{i=1}^{n} \left(\vec{P}_U \left(\, o_1 \cdots o_{k-1} \right)[i] \times \delta(q_i, q_j) \right) \times \beta(q_j, o_k) \ .$$
3. Termination:
$$\Pr(\, o_1 o_2 \cdots o_k \mid U) = \sum_{i=1}^{n} \vec{P}_U(\, o_1 o_2 \cdots o_k)[i] \ .$$

The first step of the algorithm computes the state distribution vector of a string composed of a single symbol from the alphabet (strings of length one). This step clearly takes n multiplications to produce the state distribution vector since the value of each element in the vector is computed by means of a single product. The second step recursively computes the state distribution vector of a string of length $k > 1$ from the state distribution vector of its $(k-1)$ length prefix. Each of the n elements of the state distribution vector in this step requires $(n+1)$ multiplications and $(n-1)$ sums to be performed, hence a total of $n \times ((n+1)+(n-1)) = 2n^2$ operations are carried out in step 2 for each iteration. In computing the generating probability of a string of length k, step 2 of the algorithm would be iterated $(k-1)$ times performing a total of $(k-1)(2n^2)$ operations.

The termination step simply sums up all the elements of the state distribution vector of the input string to obtain its generating probability. This step requires $(n-1)$ sums to be carried out. The algorithm then performs a total of $n + (k-1)(2n^2) + (n-1) = (2k-2)n^2 + 2n - 1 = \mathcal{O}(k \times n^2)$ operations, a notable improvement over the previous approach.

For convenience, in the remainder of this work, a version of the Forward Algorithm (without the termination step) using the matrix notation of Sec. 1.2.2 will be utilized in computing state distribution vectors:

$$1. \quad \vec{P}_U(\,\sigma) \;\; = \vec{p} \cdot \vec{D}_\sigma \,, \qquad\qquad \forall(\,\sigma \in \Sigma)\,, \tag{1.4a}$$

$$2. \quad \vec{P}_U(x\sigma) = \vec{P}_U(x) \cdot \vec{M} \cdot \vec{D}_\sigma \,, \qquad\qquad \forall\left(\,\sigma \in \Sigma\,,\; x \in \Sigma^+\right). \tag{1.4b}$$

Chapter 2:
Supervised HMM Learning

The teacher-student learning model (Angluin 1988), consists of an oracle (a teacher or expert), correctly answering the learning algorithm (learner) queries about a target concept. A concept c is defined as any subset of a given domain $\mathcal{X}$. A concept class $\mathcal{C}$ is a set of concepts. The learner's task consists of presenting a hypothesis $h \subseteq \mathcal{X}$ that matches the target concept exactly. The learner has to output such hypothesis in time polynomial in the size of its input and target concept representation. Additionally, the number of queries formulated by the learner must be bounded by a polynomial function of the size of the target concept.

Many different types of queries have been proposed within this framework such as equivalence, membership, subset, and superset queries, each requiring the availability of different kinds of oracles. In an *equivalence query*, the learner presents the oracle an hypothesis h and the oracle answers 'yes' if the hypothesis matches the target concept ($h = c$) or provides the learner with a 'counterexample', an instance $x \in \mathcal{X}$ belonging to the symmetric difference ($h \oplus c$) of h and c, otherwise. In a *membership query*, the input to the oracle is an instance $x \in \mathcal{X}$ and the output is 'yes', if $x \in c$, or 'no' if $x \notin c$. A *subset query* has for input an hypothesis h and the oracle returns 'yes' if $h \subseteq c$, or an instance $x \in (h - c)$ otherwise. In a *superset query* the learner presents the oracle a hypothesis h, and the oracle returns 'yes' if $h \supseteq c$, or an instance $x \in (c - h)$. A general problem in Computational Learning Theory, lies in determining, for each concept class, a minimal set of query combinations that allows the learner to learn the class in an efficient manner. It has been shown (Angluin 1988), for example, that DFAs cannot be efficiently learned by using exclusively either membership or equivalency queries, but that polynomial learning can be achieved by a learner combining both types of queries (Angluin 1987).

Membership queries alone are not sufficient to efficiently learn PAs either (Tzeng 1992), where a membership oracle returns the accepting probability of a queried strings. By extension, HMMs cannot be efficiently learn using a generating probability oracle, requiring the use of a stronger oracle. Our HMM learning algorithm uses an SD oracle as a teacher.

2.1 The SD Oracle

When supplied a string x of symbols as input by the algorithm, the SD oracle returns the state distribution vector $\vec{P}_U(x)$ associated with the string x. As shown previously, state distribution vectors can be computed in polynomial time by using the Forward Algorithm described in Sec. 1.2.4. Although a state distribution vector, carries more information than just the generating probability —the generating probability is actually its vector sum, it will be shown in Chapter 3 that the information carried by the SD oracle is minimal in the sense that it is not sufficient for learning HMM parameters without the ability to query specific strings. Namely, the consistency problem for HMMs, using state distribution vectors as training examples belongs to the class of NP-Complete problems.

2.2 The Supervised Learning Algorithm

The algorithm proposed, shown in Fig. 2.1, learns the ISPD, TPD, and DPD of a first order HMM, when given as input the HMM state set Q, and display alphabet Σ, and provided with access to an SD oracle for the target HMM.

In order to learn the TPD the algorithm attempts to find a basis $\vec{B}$ of linearly independent state distribution vectors, where each row $\vec{B}[i]$ of the matrix $\vec{B}$ is the state distribution vector $\vec{P}_U(x_i)$ of a string $x_i \in \Sigma^+$ in the target HMM (lines 1–16). These state distribution vectors are obtained by querying the SD oracle, represented in the algorithm by the function SD().

Algorithm *SupLearnHMM*(Q, Σ)

1. $StringQueue \longleftarrow$ EMPTY(Queue) ;
2. for each $\sigma \in \Sigma$ do
3. $\quad StringQueue \longleftarrow StringQueue \cup \{\sigma\}$;
4. end ;
5. $\vec{B} \longleftarrow$ EMPTY(Matrix) ;
6. while ($StringQueue$ not EMPTY) and (RANK$(\vec{B}) < |Q|$) do
7. $\quad x \longleftarrow$ FIRST($StringQueue$) ;
8. $\quad StringQueue \longleftarrow StringQueue - \{x\}$;
9. $\quad \vec{d}_x \longleftarrow$ SD(x) ;
10. $\quad$ if $\vec{d}_x \notin$ SPAN$(\vec{B})$ then
11. $\quad\quad \vec{B} \longleftarrow$ APPEND_ROW$(\vec{d}_x)$;
12. $\quad\quad$ for each $\sigma \in \Sigma$ do
13. $\quad\quad\quad StringQueue \longleftarrow StringQueue \cup \{x\sigma\}$;
14. $\quad\quad$ end ;
15. $\quad$ end ;
16. end ;
17. $\vec{p} \longleftarrow \vec{0}_n$;
18. for each $\sigma \in \Sigma$ do
19. $\quad \vec{p} \longleftarrow \vec{p} + SD(\sigma)$;
20. $\quad \vec{W}_\sigma \longleftarrow$ EMPTY(Matrix) ;
21. $\quad$ for each $\vec{d}_x \in \vec{B}$ do
22. $\quad\quad \vec{W}_\sigma \longleftarrow$ APPEND_ROW(SD$(x\sigma)$) ;
23. $\quad$ end ;
24. end ;
25. $\vec{W} \longleftarrow \sum_{\sigma \in \Sigma} \vec{W}_\sigma$;
26. solve for $\vec{M}$ the matrix system:

$$\begin{array}{rcl} \vec{B} \cdot \vec{M} & = & \vec{W} \\ \vec{M} \cdot \vec{1}_n^T & = & \vec{1}_n^T \\ \vec{M} & \geq & \vec{0}_{n \times n} \end{array} ;$$

27. solve for the matrices D_σ the following system of matrix equations:

$$\left.\begin{array}{rcl} \vec{B} \cdot \vec{M} \cdot \vec{D}_\sigma & = & \vec{W}_\sigma \\ \vec{D}_\sigma & \geq & \vec{0}_{n \times n} \end{array}\right\} \forall (\sigma \in \Sigma)$$
$$\sum_{\sigma \in \Sigma} \vec{D}_\sigma = \vec{I}_n \ ;$$

28. if solutions were found for $\vec{M}$, and each $\vec{D}_\sigma$ then:
29. $\quad$ return $(Q, \Sigma, \vec{M}, \left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}, \vec{p})$;
30. else return (not exists) ;

Figure 2.1: Algorithm SupLearnHMM to learn the parameters of an HMM.

The algorithm then proceeds by producing a family of $|\Sigma|$ matrices $\vec{W}_\sigma$ —one for each $\sigma \in \Sigma$, where row $\vec{W}_\sigma[i]$ of $\vec{W}_\sigma$ is generated by querying the SD oracle on the state distribution vector of the string $x_i\sigma$, such that x_i is the string that has row $\vec{B}[i]$ of $\vec{B}$ as its state distribution vector $\vec{P}_U(x_i)$ (lines 18–24). A matrix $\vec{W}$ is then computed as the sum of all the matrices $\vec{W}_\sigma$ (line 25).

Given that each row of $\vec{B}$ is the state distribution vector of a string x and the corresponding row of $\vec{W}_\sigma$ is the state distribution vector of the suffix string $x\sigma$, the following equation holds for each corresponding row of $\vec{B}$ and $\vec{W}_\sigma$:

$$\vec{P}_U(x) \cdot \vec{M} \cdot \vec{D}_\sigma = \vec{P}_U(x\sigma)$$

And therefore:

$$\vec{B} \cdot \vec{M} \cdot \vec{D}_\sigma = \vec{W}_\sigma \tag{2.1}$$

Summing up (2.1) over all $\sigma \in \Sigma$:

$$\begin{aligned} \vec{B} \cdot \vec{M} \cdot \sum_{\sigma\in\Sigma} \vec{D}_\sigma &= \sum_{\sigma\in\Sigma} \vec{W}_\sigma \\ \vec{B} \cdot \vec{M} \cdot \vec{I}_n &= \vec{W} \\ \vec{B} \cdot \vec{M} &= \vec{W} \ . \end{aligned} \tag{2.2}$$

The algorithm requires solving the matrix system in (2.2) for $\vec{M}$ in order to obtain the TPD (line 26). The additional equations shown in line 26 are included to ensure any solution for $\vec{M}$ is stochastic.

The ISPD $\vec{p}$ is computed in (lines 17–19) as the sum of the state distribution vectors of all the strings of length one in the alphabet as shown in (2.3):

Summing (1.4a) over all $\sigma \in \Sigma$:

$$\begin{aligned} \sum_{\sigma\in\Sigma} \vec{P}_U(\,\sigma) &= \vec{p} \cdot \sum_{\sigma\in\Sigma} \vec{D}_\sigma \\ &= \vec{p} \cdot \vec{I}_n = \vec{p} \ . \end{aligned} \tag{2.3}$$

Finally, the algorithm computes the DPD by solving (2.1) for each of the $\vec{D}_\sigma$ matrices (line 27).

2.2.1 Correctness

If a full basis $\vec{B}$ of n linearly independent state distribution vectors is found, the algorithm returns the ISPD, TPD, and DPD of the target HMM U^*. Otherwise, if a solution to the system of line 26 is obtained but the matrix $\vec{B}$ has rank less than n, the parameters of the HMM U learned may not be those of the target HMM U^*. However, as stated in Theorem 2.1 below concerning the correctness of the algorithm, for every string $x \in \Sigma^+$, the state distribution vectors associated with x in the HMMs U and U^* are identical.

Theorem 2.1. *Let $U^* = (Q, \Sigma, \delta, \beta, \rho)$ be a target* HMM *and U be the corresponding* HMM *learned by the algorithm in Fig. 2.1. Then for all $x \in \Sigma^+, \vec{P}_U(x) = \vec{P}_{U^*}(x)$.*

Proof. Let $S = \{x_1, \ldots, x_r\} \subseteq \Sigma^+$ be the set of $r \leq |Q|$ strings that have for state distribution vectors the rows $\{\vec{B}[1], \ldots, \vec{B}[r]\}$ of the basis $\vec{B}$ found by the learning algorithm —i.e. for $i = 1, \ldots, r : \vec{B}[i] = \vec{P}_U(x_i)$.

Let $S_0 = S$ and $S_k = \{xy : x \in S, \text{ and } |y| = k\} \cup \{\sigma y : \sigma \in \Sigma, \sigma \notin S, \text{ and } |y| = k-1\}$, the set of suffixes of the strings x in the set S of length $|x| + k$ together with the strings of length k whose first symbol is not a string in S. Note that: $\bigcup_{k=0}^{\infty} S_k = \Sigma^+$.

Theorem 2.1 is proven by induction on k:

- Base step : for $x \in S_0 \cup S_1$ it follows from the algorithm that $\vec{P}_U(x) = \vec{P}_{U^*}(x)$.
- Inductive step: Assuming $\vec{P}_U(x) = \vec{P}_{U^*}(x), \forall (x \in S_k)$.

  Consider the string $x\sigma \in S_{k+1}$:

$$\begin{aligned} \vec{P}_U(x\sigma) &= \vec{P}_U(x) \cdot \vec{M} \cdot \vec{D}_\sigma \\ &= \vec{P}_{U^*}(x) \cdot \vec{M} \cdot \vec{D}_\sigma \ . \end{aligned}$$

Since the matrix $\vec{B}$ spans the set $\{\vec{P}_{U^*}(x) : x \in \Sigma^+\}$, the vector $P_{U^*}(x)$ can be written as a linear combination of the rows $\vec{B}[i]$ of $\vec{B}$:

$$\begin{aligned}\vec{P}_U(x\sigma) &= \left(\sum_{i=1}^{r} a_i \vec{B}[i]\right) \cdot \vec{M} \cdot \vec{D}_\sigma \\ &= \sum_{i=1}^{r} a_i \left(\vec{B}[i] \cdot \vec{M} \cdot \vec{D}_\sigma\right) \quad .\end{aligned}$$

From the matrix system of line 27 of the algorithm, as per (2.1):

$$\begin{aligned}\vec{P}_U(x\sigma) &= \sum_{i=1}^{r} a_i \left(\vec{B}[i] \cdot \vec{M}^* \cdot \vec{D}^*_\sigma\right) \\ &= \sum_{i=1}^{r} \left(a_i \vec{B}[i]\right) \cdot \vec{M}^* \cdot \vec{D}^*_\sigma \\ &= \vec{P}_{U^*}(x) \cdot \vec{M}^* \cdot \vec{D}^*_\sigma \\ &= \vec{P}_{U^*}(x\sigma) \quad .\end{aligned}$$

□

2.2.2 Complexity

Theorem 2.2. *Algorithm* SupLearnHMM *has polynomial sample and time complexities in* $n = |Q|$ *and* $m = |\Sigma|$ *.*

Proof. The first part of the algorithm (lines 1–16) is dominated by the computational cost of querying the SD oracle and testing whether each state distribution vector obtained from the oracle adds to the rank of the matrix $\vec{B}$. The algorithm parses a lexicographical tree of strings in breath first search order. The first level nodes are the symbols of Σ, the second level nodes are the strings of exactly two symbols, etc. Once a string x is encountered whose state distribution vector is already in the span of $\vec{B}$, the subtree of strings rooted at x (all suffixes of x) is eliminated from future parsing. This implies that the deepest tree level reachable by the algorithm while parsing is n (strings of length n).

Let l_i represent the number of (linearly independent) state distribution vectors appended to $\vec{B}$ during the level i, $(i = 1, \ldots, n)$ parse. Then since $\vec{B}$ can at most have n linearly independent rows:

$$\sum_{i=1}^{n} l_i \leq n \ .$$

Once parsing level $k+1$, only the m one-symbol suffixes to each of the l_k strings whose linearly independent state distribution vectors were incorporated to $\vec{B}$ in the previous level (level k) still remain in the queue for consideration. The number of queries to the SD oracle (sample complexity) performed by the algorithm during the parsing is therefore bound by:

$$\sum_{i=1}^{n} (l_i \times m) = m \times \sum_{i=1}^{n} l_i \leq m \times n \ .$$

As explained in Sec. 1.2.4, the state distribution vectors can be computed by the Forward Algorithm in $\mathcal{O}(n^3)$(for strings of length n). Evaluating whether a state distribution vector $\vec{v}$ is in the span of the matrix $\vec{B}$ (line 16) involves computing the rank of the augmented matrix resulting from appending $\vec{v}$ as a new row of $\vec{B}$. The rank computation can be performed using singular value decomposition which for an $(n \times n)$ matrix involves $\mathcal{O}(n^3)$ multiplications (Chan 1982). The total complexity for the first section of the algorithm (lines 1–16) is therefore $(m \times n)(n^3 + n^3) = \mathcal{O}(m \times n^4)$ which is polynomial in m and n.

In lines 17–24, the algorithm queries the state distribution vector of all of the m one-symbol suffixes for the strings whose state distribution vectors form the rows of $\vec{B}$. This produces a maximum of $(m \times n)(n+1)^3$ operations (worst case scenario $\vec{B}$ has n rows with some of them being state distribution vectors corresponding to strings of length n).

The last section of the algorithm, involves finding feasible solutions (not necessarily optimal) to the positive matrix systems in lines 26, and 27, which can be solved using

linear programming techniques in polynomial time also (Karmarkar 1984). It is then concluded that algorithm produces an answer in polynomial time in n and m as well. □

2.2.3 Simulation Results

A simulation experiment has been devised in order to test the viability of the learning algorithm. The experiment consisted of constructing 1,000 HMMs for each of several combinations of $n = |Q|$ and $m = |\Sigma|$. For each target HMM, the values of its ISPD, TPD, and DPD were randomly generated. The learning algorithm was then used to obtain the basis $\vec{B}$ of state distribution vectors and the number of queries to the SD oracle performed by the algorithm was recorded.

Table 2.1 shows the average number of queries to the SD oracle required by the learning algorithm in order to obtain a full basis $\vec{B}$ of linearly independent state distribution vectors. As seen from the table, the average number of queries performed is nearly the size n of the HMM state set Q.

It was also observed throughout the experiment that in less than one percent of the HMMs constructed, the matrix $\vec{B}$ of linearly independent state distribution vectors obtained by the learning algorithm had a rank smaller than $n = |Q|$.

Table 2.1: Average number of queries performed to obtain a basis of linearly independent distribution vectors corresponding to a randomly generated HMM of $n = |Q|$ states and $m = |\Sigma|$ display symbols.

		n							
		5	6	7	10	15	20	25	30
m	2	5.02	6.02	7.02	10.03	15.07	20.75	29.34	43.83
	3	5.04	6.05	7.04	10.06	15.08	20.20	25.74	33.17
	4	5.08	6.08	7.06	10.08	15.11	20.13	25.38	31.20
	5	5.13	6.12	7.09	10.13	15.15	20.23	25.44	30.67
	10	5.29	6.35	7.37	10.34	15.55	20.68	26.08	31.53

2.2.4 Conditions for the Existence of a Full Basis of State Distribution Vectors

There are a number of conditions on the parameters of a target HMM that although not necessary, are indeed sufficient to guarantee the existence of a full basis of state distribution vectors from the HMM. Such conditions are established and proved by Lemma 2.4, and subsequent Theorems 2.3 and 2.4.

Several definitions and lemmas from elementary linear algebra that are required in order to state the conditions and prove the theorems stating the existence of a basis, are given first.

Definition 2.1. A non-zero row vector $\vec{v}$ is a *left eigenvector* of a square matrix $\vec{A}$ if and only if there is a scalar λ, such that $(\vec{v} \cdot \vec{A}) = \lambda \times \vec{v}$.

Note: The usual right —i.e. column— eigenvectors will be referred from now on simply as 'eigenvectors'. When referring to left —i.e. row eigenvectors— the term 'left eigenvector' will be explicitly used.

Lemma 2.1. *A non-zero row vector $\vec{v}$ is a left eigenvector of a square matrix $\vec{A}$ corresponding to the eigenvalue λ if and only if $\vec{v}^{\,T}$ is an eigenvector of $\vec{A}^{\,T}$ corresponding to λ (i.e. $(\vec{A}^{\,T} \cdot \vec{v}^{\,T}) = \lambda \times \vec{v}^{\,T}$).*

Proof. $(\vec{v} \cdot \vec{A}) = \lambda \times \vec{v} \Longleftrightarrow (\vec{v} \cdot \vec{A})^{\,T} = (\lambda \times \vec{v})^{\,T} \Longleftrightarrow \vec{A}^{\,T} \cdot \vec{v}^{\,T} = \lambda \times \vec{v}^{\,T}$. □

Definition 2.2. The *spectrum* of a square matrix $\vec{A}$ is the set containing all the eigenvalues of $\vec{A}$.

Lemma 2.2. *The eigenvectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ of a matrix $\vec{A}$ corresponding to distinct eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are linearly independent.*

Lemma 2.3. *Let $\vec{A}$ be an $(n \times n)$ matrix having a spectrum of n distinct eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, then $\vec{A}$ has the following Spectral (or Eigen) Decomposition:*

$$\vec{A} = \vec{R} \cdot \vec{\Lambda} \cdot \vec{R}^{-1} \ ,$$

where:

- $\vec{\Lambda}$ *is the* $(n \times n)$ *diagonal matrix whose main diagonal contains the eigenvalues of* $\vec{A}\left(\forall(1 \leq i \leq n) : \vec{\Lambda}[i,i] = \lambda_i\right)$*:*

$$\vec{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix} .$$

- $\vec{R} = [\vec{v}_1\, \vec{v}_2 \cdots \vec{v}_n]$ *is the* $(n \times n)$ *matrix whose columns are linearly independent eigenvectors of* $\vec{A}$*, such that* $\vec{v}_i$ *is the eigenvector corresponding to the eigenvalue* λ_i *of* $\vec{A}$*,* $i = 1 \ldots, n$*.*

- $\vec{R}^{-1}$ *is the matrix inverse of* $\vec{R}$*. The rows of* $\vec{R}^{-1}$ *are the left eigenvectors of* $\vec{A}$*, such that for* $i = 1, \ldots, n$ *, row* $\vec{R}^{-1}[i]$ *is the left eigenvector corresponding to the eigenvalue* λ_i *.*

Lemmas 2.2 and 2.3 above are well-known results from elementary linear algebra. See (Hohn 1958) and (Goode 1991) for details of their proofs.

Lemma 2.4. *Let* $\vec{W}$ *be an* $(n \times n)$*-dimensional matrix such that the spectrum of* $\vec{W}$ *consists of* n *distinct eigenvalues. Let* $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ *and* $(\vec{R} \cdot \vec{\Lambda} \cdot \vec{R}^{-1})$ *be the spectrum and the spectral decomposition, respectively, of* $\vec{W}$*.*

Let $\vec{u}$ *be an* n*-dimensional row vector,* $\vec{u} = [k_1\ k_2 \cdots k_n] \cdot \vec{R}^{-1}$ *such that* $\forall(1 \leq i \leq n)$ *:* $k_i \neq 0$ *.*

Then, the vector set $\left\{\vec{u}, (\vec{u} \cdot \vec{W}), (\vec{u} \cdot \vec{W}^2), \ldots, (\vec{u} \cdot \vec{W}^{n-1})\right\}$ *is linearly independent.*

Proof. Let $c_1, c_2, \ldots, c_n$ be any constants such that:

$$c_1 \times \vec{u} + c_2 \times (\vec{u} \cdot \vec{W}) + c_3 \times (\vec{u} \cdot \vec{W}^2) + \cdots + c_n \times (\vec{u} \cdot \vec{W}^{n-1}) = \vec{0}_n^T \ .$$

Let $P(x)$ be the polynomial:

$$P(x) = c_1 \times x^0 + c_2 \times x^1 + c_3 \times x^2 + \cdots + c_n \times x^{n-1} \ .$$

Then:

$$\begin{aligned}
& c_1 \times \vec{u} + c_2 \times (\vec{u} \cdot \vec{W}) + c_3 \times (\vec{u} \cdot \vec{W}^2) + \cdots + c_n \times (\vec{u} \cdot \vec{W}^{n-1}) && = \vec{0}_n^T \\
\iff & \vec{u} \cdot \left(c_1 \times \vec{I}_n + c_2 \times \vec{W} + c_3 \times \vec{W}^2 + \cdots + c_n \times \vec{W}^{n-1} \right) && = \vec{0}_n^T \\
\iff & \vec{u} \cdot P(\vec{W}) && = \vec{0}_n^T \\
\iff & \left([k_1 \ k_2 \cdots k_n] \cdot \vec{R}^{-1} \right) \cdot P(\vec{R} \cdot \vec{\Lambda} \cdot \vec{R}^{-1}) && = \vec{0}_n^T \\
\iff & \left([k_1 \ k_2 \cdots k_n] \cdot \vec{R}^{-1} \right) \cdot \left(\vec{R} \cdot P(\vec{\Lambda}) \cdot \vec{R}^{-1} \right) && = \vec{0}_n^T \\
\iff & [k_1 \ k_2 \cdots k_n] \cdot \left(\vec{R}^{-1} \cdot \vec{R} \right) \cdot P(\cdot \vec{\Lambda}) \cdot \vec{R}^{-1} && = \vec{0}_n^T \\
\iff & [k_1 \ k_2 \cdots k_n] \cdot \vec{I}_n \cdot P(\cdot \vec{\Lambda}) \cdot \vec{R}^{-1} && = \vec{0}_n^T \\
\iff & [k_1 \ k_2 \cdots k_n] \cdot P(\cdot \vec{\Lambda}) \cdot \vec{R}^{-1} && = \vec{0}_n^T \\
\iff & [k_1 \ k_2 \cdots k_n] \cdot \begin{bmatrix} P(\lambda_1) & 0 & \cdots & 0 \\ 0 & P(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & P(\lambda_n) \end{bmatrix} \cdot \vec{R}^{-1} && = \vec{0}_n^T \\
\iff & [k_1 P(\lambda_1) \ k_2 P(\lambda_2) \cdots k_n P(\lambda_n)] \cdot \vec{R}^{-1} && = \vec{0}_n^T
\end{aligned}$$

Since the rows of $\vec{R}^{-1}$ are linearly independent vectors:

$$\begin{aligned}
& [k_1 P(\lambda_1) \ k_2 P(\lambda_2) \cdots k_n P(\lambda_n)] \cdot \vec{R}^{-1} && = \vec{0}_n^T \\
\iff & k_1 P(\lambda_1) = k_2 P(\lambda_2) = \cdots = k_n P(\lambda_n) && = 0 \\
\iff & P(\lambda_1) = P(\lambda_2) = \cdots = P(\lambda_n) = 0 \qquad (\text{since } k_i \neq 0, \forall (1 \leq i \leq n)) \, .
\end{aligned}$$

$P(\lambda_1) = P(\lambda_2) = \cdots = P(\lambda_n) = 0$ implies that the polynomial $P(x)$ is either null ($P(x) = 0$), or it has $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ as n distinct roots which contradicts the fact that $P(x)$ has at most degree $n-1$. Therefore, $P(x) = 0$ and hence $c_1 = c_2 = \cdots = c_n = 0$. Consequently the n vectors in $\left\{\vec{u}, (\vec{u} \cdot \vec{W}), (\vec{u} \cdot \vec{W}^2), \ldots, (\vec{u} \cdot \vec{W}^{n-1})\right\}$ are linearly independent and form a basis. □

Theorem 2.3. *Let* $U^{=}(Q, \Sigma, \vec{M}, \left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}, \vec{p})$ *be an* HMM *such that:*

1. $\exists \sigma$, $\sigma \in \Sigma$, *such that the spectrum of the matrix* $(\vec{M} \cdot \vec{D}_\sigma)$ *consists of* n *distinct eigenvalues.*

2. *The row vector* $(\vec{p} \cdot \vec{D}_\sigma)$ *can be expressed as a linear combination of the* n*-linearly independent left eigenvectors of* $(\vec{M} \cdot \vec{D}_\sigma)$ *with no null coefficients in the linear combination.*

Then, the set of state distribution vectors $\left\{\vec{P}_U(\sigma), \vec{P}_U(\sigma\sigma), \ldots, \vec{P}_U(\sigma^n)\right\}$ *is linearly independent.*

Proof. Since $\forall (1 \leq i \leq n) : \vec{P}_U(\sigma^i) = (\vec{p} \cdot \vec{D}_\sigma) \cdot (\vec{M} \cdot \vec{D}_\sigma)^{i-1}$:

$$\begin{aligned}
\vec{P}_U(\sigma) &= (\vec{p} \cdot \vec{D}_\sigma) \\
\vec{P}_U(\sigma\sigma) &= (\vec{p} \cdot \vec{D}_\sigma) \cdot (\vec{M} \cdot \vec{D}_\sigma) \\
&\vdots \\
\vec{P}_U(\sigma^n) &= (\vec{p} \cdot \vec{D}_\sigma) \cdot (\vec{M} \cdot \vec{D}_\sigma)^{n-1} \ .
\end{aligned}$$

Hence, taking $\vec{u} = (\vec{p} \cdot \vec{D}_\sigma)$, and $\vec{W} = (\vec{M} \cdot \vec{D}_\sigma)$ by Lemma 2.4, the vectors

$$\left\{\vec{P}_U(\sigma), \vec{P}_U(\sigma\sigma), \ldots, \vec{P}_U(\sigma^n)\right\}$$

are linearly independent and form a basis. □

Theorem 2.3 can be generalized as stated by Theorem 2.4 below.

Theorem 2.4. *Let* $U^{=}(Q, \Sigma, \vec{M}, \left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}, \vec{p})$ *be an* HMM *such that:*

- $\exists \sigma, \sigma \in \Sigma$*, such that the spectrum of the matrix* $(\vec{M} \cdot \vec{D}_\sigma)$ *consists of* n *distinct eigenvalues.*
- *The row vector* $\vec{p}$ *can be expressed as a linear combination of the* n*-linearly independent left eigenvectors of* $(\vec{M} \cdot \vec{D}_\sigma)$ *with no null coefficients in the linear combination.*

Let $S = S_1 \cup S_2 \cup \cdots \cup S_m$ *such that:*

$$\forall (1 \leq i \leq m) : S_i = \left\{\vec{P}_U(\sigma_i), \vec{P}_U(\sigma_i\sigma), \vec{P}_U(\sigma_i\sigma^2), \ldots, \vec{P}_U(\sigma_i\sigma^{n-1})\right\} \ .$$

Then, the set of $(m \times n)$*,* n*-dimensional state distribution vectors* S *above contains a basis of* n *linearly independent vectors.*

Proof. From the hypothesis, there exists $\sigma \in \Sigma$, say σ_1, such that $(\vec{M} \times \vec{D}_{\sigma_1})$ posses a full set $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ of n distinct eigenvalues.

Now for each $i = 1, \ldots, m$, consider the following matrices $\vec{B}_i$ whose j^{th} row is the j^{th} vector in S_i:

$$\vec{B}_i[j] = P_U(\sigma_i\sigma_1^{j-1}) = \vec{p} \cdot \vec{D}_{\sigma_i} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{j-1} \ .$$

Let $\vec{B} = \sum_{i=1}^{m} \vec{B}_i$. Then, for all $j = 1, \ldots, n$:

$$\begin{aligned}
\vec{B}[j] &= \sum_{i=1}^{m} \vec{B}_i[j] \\
&= \sum_{i=1}^{m} \vec{p} \cdot \vec{D}_{\sigma_i} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{j-1} \\
&= \vec{p} \cdot \left(\sum_{i=1}^{m} \vec{D}_{\sigma_i}\right) \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{j-1} \\
&= \vec{p} \cdot \vec{I}_n \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{j-1} \\
&= \vec{p} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{j-1} \ .
\end{aligned}$$

Taking $\vec{u} = \vec{p}$, and $\vec{W} = (\vec{M} \cdot \vec{D}_{\sigma_1})$, and given that the rows of $\vec{B}$ are:

$$\begin{aligned}
\vec{B}[1] &= \vec{p} \ , \\
\vec{B}[2] &= \vec{p} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1}), \\
\vec{B}[3] &= \vec{p} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^2, \\
&\vdots \\
\vec{B}[n] &= \vec{p} \cdot (\vec{M} \cdot \vec{D}_{\sigma_1})^{n-1} \ .
\end{aligned}$$

By Lemma 2.4, the n rows of matrix $\vec{B}$ are linearly independent vectors and therefore the set S of state distribution vectors contains a basis. □

Chapter 3:
HMM Consistency Problem Using State Distribution Vectors

In order to demonstrate that the information provided by the SD oracle is not too strong a requirement, and that the SD oracle is in fact necessary for polynomial-time HMM learning using state distribution vector information, Theorem 3.1 proves that the consistency problem for HMMs using state distribution vectors —such as those carried by the SD oracle, where the ability to query the state distribution vectors of specific strings is inhibited, is NP-Complete. The consistency problem for HMM using state distribution vectors is defined next.

Definition 3.1. Given a dataset T^h of training examples of the form $\langle\, x\,, \vec{v}\,\rangle$ where x is a string from some alphabet Σ, and $\vec{v}$ is an n-dimensional state distribution vector associated with the string x, the *Consistency Problem* for HMM using state distribution vectors is to determine whether there exists an HMM $U = (Q, \Sigma, \delta, \beta, \rho)$ consistent with T^h —i.e. $|\,Q| = n$ and for each $\langle\, x\,, \vec{v}\,\rangle \in T^h$, $\vec{P}_U(x) = \vec{v}$.

Theorem 3.1 (NP-Completeness)**.** *The consistency problem for* HMM *using state distribution vectors is NP-Complete.*

The proof of Theorem 3.1 proceeds from Tzeng's reduction (Tzeng 1992) of the SAT' problem (Gold 1978) —satisfiability of a set C of boolean clauses such that every clause in C involves all positive or all negative literals only— to a Deterministic Finite Automata (DFA) consistency problem. Tzeng defines a set T of examples of the form $\langle\, x\,, q_i\,\rangle$ for a DFA, and proves that there exists a DFA A consistent with T if and only if the set of clauses C is satisfiable. Theorem 3.1 is proven by constructing an HMM U and a set T^h of examples of the form described in Definition 3.1 such that U is consistent with T^h if and only if A is consistent with T.

In order to construct the HMM U and example dataset T^h, first the DFA A, and dataset T corresponding to the SAT' reduction are transformed into a DFA $A^@$ and dataset $T^@$ as described in Definitions 3.4 and 3.5 respectively. Theorem 3.2 proves that the DFAs A and $A^@$ are equivalent in the sense that A is consistent with T if and only if $A^@$ is consistent with $T^@$. Fig. 3.1 shows a sketch of the proof sequence.

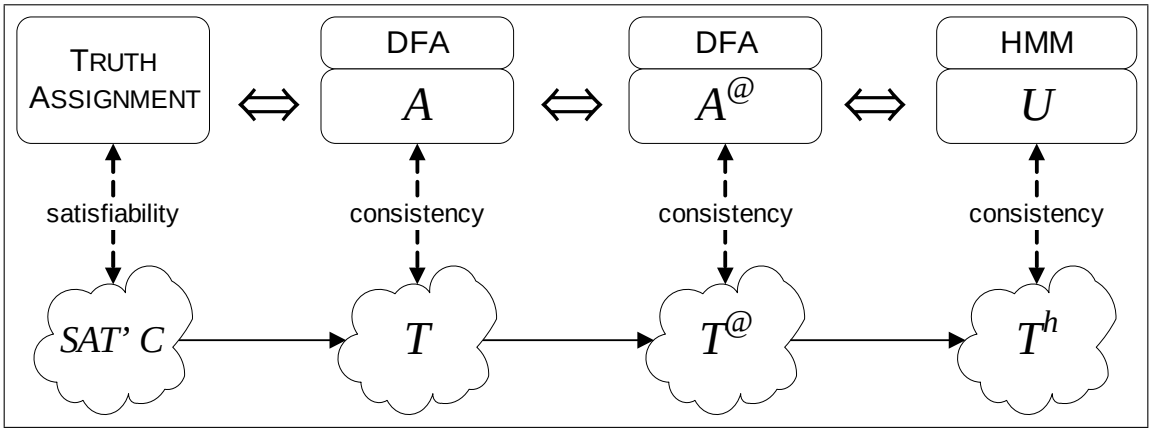

Figure 3.1: Steps involved in the proof of Theorem 3.1.

Definition 3.7 defines the training dataset T^h, and finally Theorems 3.3 and 3.4 respectively prove the forward and backward directions of Theorem 3.1. Tzeng's reduction of the SAT' problem to a DFA consistency problem is described first.

3.1 SAT' Reduction to DFA

Definition 3.2. Given a dataset T of training examples of the form $\langle x_i, q_i \rangle$ where x_i is a string from some alphabet Σ, and q_i is a state from a state set Q, the *Consistency Problem* for DFA is to determine whether there exists a DFA $A = (Q, \Sigma, \delta, q_1, \cdot)$ consistent with T —i.e. for each $\langle x_i, q_i \rangle \in T$, $\delta(q_1, x_i) = q_i$ and $Q = \{q_i : \exists \langle x_i, q_i \rangle \in T\}$.

Let $C = \{c_1, \ldots, c_r\}$ be a set of clauses over a set of propositional variables $V = \{v_1, \ldots, v_l\}$, such that each clause c_i is either positive (contains only positive literals) or negative (contains only negative literals).

Let A_V and $A_{C_i}, i = 0, \ldots, l$ be the tree-like automata of Fig. 3.2. A_V has state set Q_V with state q_v as the root and leaf states $\{q_{v_1}, \ldots, q_{v_{l'}}\}$ where $l' = 2^{\lceil \log l \rceil}$. Each leaf state $q_{v_i}, i = 1, \ldots, l$, corresponds with the variable $v_i \in V$. The height of A_V is $\lceil \log l \rceil$.

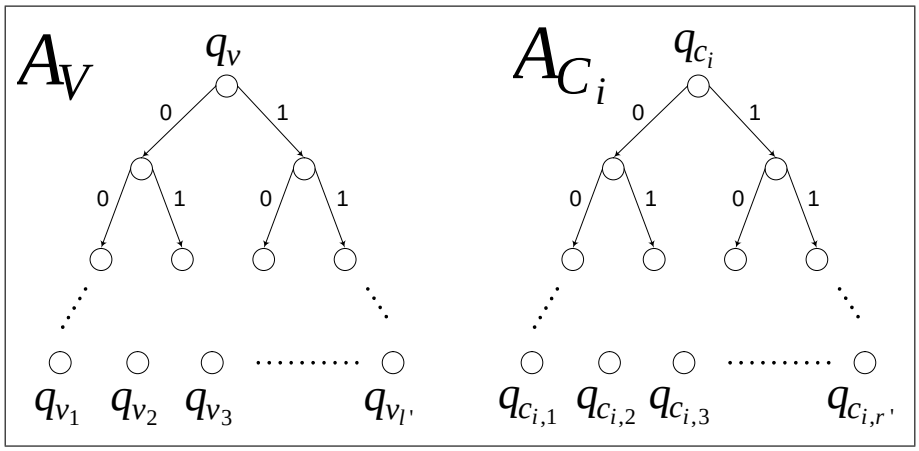

Figure 3.2: Tree-like automata A_V and $A_{C_i}, i = 0, \ldots, l$.

The family of automata $A_{C_i}, i = 0, \ldots, l$, each have a state set Q_{C_i} with q_{c_i} being the root state for each tree and $\{q_{c_i,1}, \ldots, q_{c_i,r'}\}$ being the leaf states, where $r' = 2^{\lceil \log r \rceil}$. Each leaf state $q_{c_i,j}$ corresponds to the clause $c_j \in C$. The A_{C_i} trees have height $\lceil \log r \rceil$.

Definition 3.3. Let $\delta : Q \times \Sigma \to Q$ denote a transition function.

Let $x_{v_i} \in \Sigma^+$ denote the string such that $\delta(q_v, x_{v_i}) = q_{v_i}, i = 1, \ldots, l'$.

Let $x_{c_j} \in \Sigma^+$ denote the string such that $\delta(q_{c_i}, x_{c_j}) = q_{c_i,j}, j = 1, \ldots, r'$.

Let $Q = \{q_1, q_2, q_3, q_4, q_5, q_6\} \cup Q_V \cup \left(\bigcup_{i=0}^{l} Q_{C_i} \right)$.

Let $T = T_1 \cup T_2 \cup T_3 \cup T_4 \cup T_5 \cup T_V \cup \left(\bigcup_{i=0}^{l} T_{C_i} \right)$ be a *set of transitions* of the form $\langle\, x\,, q_i \,\rangle$ —represented as $\delta(q_1, x) = q_i$ for convenience— where:

T_1: $\delta(q_1, 0) = q_v$,

$\delta(q_1, 1) = q_{c_0}$,

$\delta(q_1, 0x_{v_i}1) = \delta(q_{v_i}, 1) = q_{c_i} \qquad \forall(\, 1 \leq i \leq l)$,

$$T_2\text{: } \delta(q_1, 0x_{v_i}1x_{c_j}0) = \begin{cases} q_2 & \text{if } v_i \text{ in } c_j \ , \\ q_3 & \text{otherwise} \ , \end{cases} \qquad \forall(\, 1 \le i \le l,\ 1 \le j \le r) \ ,$$

$$T_3\text{: } \delta(q_1, 1x_{c_j}01x_{c_j}0) = q_2, \qquad \forall(\, 1 \le j \le r) \ ,$$

$$T_4\text{: } \delta(q_1, 1x_{c_j}00) = \begin{cases} q_4 & \text{if } c_j \text{ is positive} \ , \\ q_5 & \text{if } c_j \text{ is negative} \ , \end{cases} \qquad \forall(\, 1 \le j \le r) \ ,$$

$$T_5\text{: } \delta(q_i, y_i\sigma) = \delta(q_i, \sigma) = q_6 \qquad \text{where } \sigma \in \{0,1\}\,,\ \delta(q_1, y_i) = q_i\,, \text{ and } (2 \le i \le 6)\,,$$

$$\delta(q_1, 0x_{v_i}1x_{c_j}0) = \delta(q_{c_i,j}, 0) = q_6, \qquad \forall(\, 1 \le i \le l\ ,\ r+1 \le j \le r') \ ,$$

$$\delta(q_1, 0x_{v_i}1x_{c_j}1) = \delta(q_{c_i,j}, 1) = q_6, \qquad \forall(\, 0 \le i \le l\ ,\ 1 \le j \le r') \ ,$$

$$\delta(q_1, 0x_{v_i}\sigma) = \delta(q_{v_i}, \sigma) = q_6, \qquad \forall(\, l+1 \le i \le l')\ ,\ \sigma \in \{0,1\} \ ,$$

T_V: Transitions defined from A_V ,

T_{C_i}: Transitions defined from A_{C_i}, $i = 1, \ldots, l$.

According to (Tzeng 1992) there exists a DFA $A = (Q, \Sigma, \delta, q_1, \cdot)$ consistent with the set of transitions T if an only if the set of clauses C is satisfiable.

3.2 DFA Reduction to HMM

Definition 3.4. Let $A = (Q, \Sigma, \delta, q_1, F)$ be a DFA. A new DFA $A^{@} = (Q, \Sigma^{@}, \delta^{@}, q_1, F)$ is defined from A, where @ is a new symbol, @ $\notin \Sigma$, and:

i) $\Sigma^{@} = \Sigma \cup \{@\} = \{0, 1, @\}$,

ii) $$\delta^{@}(q, \sigma) = \begin{cases} \delta(q, \sigma) & \text{if } \sigma \in \Sigma, \\ q & \text{if } \sigma = @. \end{cases} \qquad \forall(q \in Q)\,.$$

Definition 3.5. Let T be a finite set of examples of the form $\langle\, x\,, q\,\rangle$, where $x \in \Sigma^*$, and $q \in Q$. A new example dataset $T^{@}$ is defined:

$$T^{@} = T \cup \{\langle\, x@\,, q\,\rangle : \langle\, x\,, q\,\rangle \in T\} \cup \{\langle\, x@y\,, q\,\rangle : \langle\, xy\,, q\,\rangle \in T\} \ . \tag{3.1}$$

Theorem 3.2. *A* DFA $A = (Q, \Sigma, \delta, q_1, F)$ *is consistent with a set of examples* T *if an only if the corresponding* DFA $A^{@} = (Q, \Sigma^{@}, \delta^{@}, q_1, F)$ *is consistent with the set* $T^{@}$ *.*

Proof. If A is consistent with T then $A^{@}$ is consistent with the examples in T as well. It suffices to prove then that $A^{@}$ is consistent with the examples in $(T^{@} - T)$, namely the examples from $T^{@}$ of the form $\langle\, x@\,, q\,\rangle$, and $\langle\, x@y\,, q\,\rangle$, where $\langle\, x\,, q\,\rangle \in T$, and $\langle\, xy\,, q\,\rangle \in T$.

For each $\langle\, x@\,, q\,\rangle \in T^{@}$:

$$\begin{aligned}
\delta^{@}(q_1, x@) &= \delta^{@}\left(\delta^{@}\left(q_1, x\right), @\right) \\
&= \delta^{@}\left(\delta\left(q_1, x\right), @\right) \\
&= \delta^{@}(q, @) \\
&= \boxed{q} \ .
\end{aligned}$$

For each $\langle\, x@y\,, q\,\rangle \in T^{@}$:

$$\begin{aligned}
\delta^{@}(q_1, x@y) &= \delta^{@}\left(\delta^{@}\left(\delta^{@}\left(q_1, x\right), @\right), y\right) \\
&= \delta^{@}\left(\delta^{@}\left(\delta\left(q_1, x\right), @\right), y\right) \\
&= \delta^{@}\left(\delta\left(q_1, x\right), y\right) \\
&= \delta\left(\delta\left(q_1, x\right), y\right) \\
&= \delta(q_1, xy) \\
&= \boxed{q} \ .
\end{aligned}$$

The proof for the backward proposition is straightforward since $T \subset T^{@}$. Therefore $A^{@}$ is consistent with the examples in T corresponding to strings that are restricted to symbols in the alphabet Σ (strings that do not contain the symbol '@'). Thus, the DFA A obtained by restricting $A^{@}$ to the alphabet Σ and transition function $\delta = \delta^{@}_{|\Sigma}$ is consistent with the examples in the set T. □

Definition 3.6. Applying now the transformation shown in Definition 3.5 to the set of examples T produced by the reduction of the SAT' problem to a DFA problem. A dataset $T^{@}$ is obtained such that:

1. $T \subseteq T^{@}$.
2. for each example $\langle\, x\,, q_i \,\rangle \in T$, the example $\langle\, x@\,, q_i \,\rangle \in T^{@}$.
3. for each example $\langle\, xy\,, q_i \,\rangle \in T$, the example $\langle\, x@y\,, q_i \,\rangle \in T^{@}$.

The number of examples in $T^{@}$ remains a polynomial function of r, the number of clauses in C, and l, the number of propositional variables. For each example $\langle\, x\,, q \,\rangle \in T$, the set $T^{@}$ incorporates $|x|$ additional examples, each produced by inserting the symbol '@' after each of the $|x|$ positions of the string x. Since for all $x \in T$, $|x| \leq (\lceil \log l \rceil + \lceil \log r \rceil)$, then $|T^{@}| \leq |T| \times (\lceil \log l \rceil + \lceil \log r \rceil)$ which is still polynomial in r, and l .
Note that $T^{@}$ contains the examples of the form $\left\langle\, 1x_{c_j}0@0\,, q_i \,\right\rangle$ where $i \in \{4, 5\}$, and $\left\langle\, 1x_{c_j}0@1x_{c_j}0\,, q_2 \,\right\rangle$, since they correspond, respectively, to the examples $\left\langle\, 1x_{c_j}00\,, q_i \,\right\rangle$ where $i \in \{4, 5\}$, and $\left\langle\, 1x_{c_j}01x_{c_j}0\,, q_2 \,\right\rangle$ from the example dataset T of the SAT' problem (see transitions T_4, and T_3 in Definition 3.3.)

Corollary 3.1. *Let $T^{@}$ be the example dataset obtained, using the transformation described in Definition 3.5, from the transition dataset T corresponding to a* SAT' *problem —as described in Definition 3.3. Then there exists a* DFA $A = (Q, \Sigma, \delta, q_1, F)$ *consistent with T if and only if there is a* DFA $A^{@} = (Q, \Sigma^{@}, \delta^{@}, q_1, F)$ *consistent with $T^{@}$.*

In order to construct a set of examples for an HMM from an example set corresponding to a DFA, additional notations will be introduced:

Definition 3.7. Let $T^{@}$ be a transition dataset for a DFA constructed from a transition dataset T according to Definition 3.5, a corresponding transition dataset T^{h} for an HMM is defined as follows:

For each example $\langle\, x\,, q_i\,\rangle \in T^{@}$:

$$\left\langle\, x0\,, \left[\!\!\left[\frac{\vec{e_i}}{3^{|x|+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \right\rangle \in T^h \tag{3.2a}$$

$$\left\langle\, x1\,, \left[\!\!\left[\vec{0}_n, \frac{\vec{e_i}}{3^{|x|+1}}, \vec{0}_n \right]\!\!\right] \right\rangle \in T^h \tag{3.2b}$$

$$\left\langle\, x@\,, \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i}}{3^{|x|+1}} \right]\!\!\right] \right\rangle \in T^h. \tag{3.2c}$$

Note that the second component of each example pair in T^h represents a state distribution vector induced by the string on the pair's first component.

The number of examples in the training set T^h is $|T^h| \leq 3 \times |T^{@}|$, since for each string x corresponding to an example in $T^{@}$, T^h incorporates examples for the suffixes $x0$, $x1$, and $x@$ some of which may be members of $T^{@}$ as well. Consequently, $|T^h|$ is polynomial in r, and l .

Theorem 3.3. *Let T be the set of* DFA *transitions corresponding to the* SAT' *reduction described in Definition 3.3. Let $T^{@}$ be the transition dataset obtained by applying the transformation of Definition 3.5 to the set T. Let T^h denote the set of* HMM *examples obtained from $T^{@}$ according to Definition 3.7.*
Then if there is a DFA $A = (Q, \Sigma, \delta, q_1, F)$ *consistent with T, then there exists an* HMM $U = (Q^h, \Sigma^{@}, \delta^h, \beta, \rho)$ *consistent with T^h.*

Proof. Since there is a DFA $A = (Q, \Sigma, \delta, q_1, F)$ consistent with T then from Theorem 3.2 the DFA $A^{@} = (Q, \Sigma^{@}, \delta^{@}, q_1, F)$ is consistent with $T^{@}$.
Let $\vec{p} = \left[\!\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1} \right]\!\!\right]$ and let $n = |\, Q|$.
Let $\vec{M}_0, \vec{M}_1, \vec{M}_{@}$ be $(n \times n)$-dimensional stochastic matrices such that:

$$\vec{M}_\sigma[i,j] = \begin{cases} 1 & \text{if } \delta^{@}(q_i, \sigma) = q_j \ , \\ 0 & \text{otherwise} \ , \end{cases} \qquad \forall(\, \sigma \in \Sigma^{@}) \ .$$

Let $Q^h = \{q_1^h, \ldots, q_n^h, q_{n+1}^h, \ldots, q_{2n}^h, q_{2n+1}^h, \ldots, q_{3n}^h\}$ be a new set of $3n$ states. The state set Q^h arises from splitting every state $q_i \in Q$ into three states $\{ q_i^h , q_{n+i}^h , q_{2n+i}^h \} \in Q^h$. Let $\vec{M}$ be a $(3n \times 3n)$ stochastic matrix such that:

$$\vec{M} = \begin{bmatrix} \frac{1}{3}\vec{M}_0 & \frac{1}{3}\vec{M}_0 & \frac{1}{3}\vec{M}_0 \\ \frac{1}{3}\vec{M}_1 & \frac{1}{3}\vec{M}_1 & \frac{1}{3}\vec{M}_1 \\ \frac{1}{3}\vec{M}_@ & \frac{1}{3}\vec{M}_@ & \frac{1}{3}\vec{M}_@ \end{bmatrix} \quad . \tag{3.3}$$

Let $\vec{D}_0$, $\vec{D}_1$, and $\vec{D}_@$ be $(3n \times 3n)$-dimensional matrices defined as:

$$\vec{D}_0 = \begin{bmatrix} \vec{I}_n & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \quad , \tag{3.4a}$$

$$\vec{D}_1 = \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{I}_n & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \quad , \tag{3.4b}$$

$$\vec{D}_@ = \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n \end{bmatrix} \quad . \tag{3.4c}$$

It is important to note that:

$$\sum_{\sigma \in \Sigma^@} \vec{D}_\sigma = \vec{D}_0 + \vec{D}_1 + \vec{D}_@ = \vec{I}_{3n} \ . \tag{3.5}$$

It will be shown that the HMM $U = (Q^h, \Sigma^@, \delta^h, \beta, \rho)$ is consistent with the dataset T^h, where:

- $\rho(q_i^h) = \vec{p}\,[i]\,, \qquad \forall(\, 1 \le i \le 3n) \ ,$
- $\beta(q_i^h, \sigma) = \vec{D}_\sigma[i,i]\,, \qquad \forall(\, 1 \le i \le 3n \ , \ \sigma \in \Sigma^@) \ ,$
- $\delta^h(q_i^h, q_j^h) = \vec{M}[i,j]\,, \qquad \forall(\, 1 \le i \le 3n \ , \ 1 \le j \le 3n) \ .$

For convenience, in the following sections, the pairs $\langle\, x_i\,, q_i\,\rangle$ of a DFA example dataset will be represented in the form $\langle\, x_i\,, \vec{e_i}\,\rangle$, where the row vector $\vec{e_i}$ is to be interpreted as the state distribution vector corresponding to the input string x_i.

Let $x_i = o_1 o_2 \ldots o_k$, $(o_i \in \Sigma^{@})$ be a string such that $\langle\, x_i\,, \vec{e_i}\,\rangle \in T^{@}$, and let $\vec{M}_{x_i} = \vec{M}_{o_1} \cdot \vec{M}_{o_2} \cdot \ldots \cdot \vec{M}_{o_k}$.

Then, since $A^{@}$ is consistent with $T^{@}$:

$$\begin{aligned}
\delta^{@}(q_1, x_i) &= \delta^{@}(q_1, o_1 o_2 \ldots o_k) \\
&= \vec{e_1} \cdot \vec{M}_{o_1} \cdot \vec{M}_{o_2} \cdot \ldots \cdot \vec{M}_{o_k} \\
&= \vec{e_1} \cdot \vec{M}_{x_i} \qquad (3.6) \\
&= \boxed{\vec{e_i}}\ . \qquad (3.7)
\end{aligned}$$

Next, it will be proven that the HMM U is consistent with the HMM transitions (3.2a), (3.2b), and (3.2c) of T^h from Definition 3.7.

It can be easily shown, by mathematical induction on k that the following matrix equations hold:

$$\begin{aligned}
\vec{P}_U(0 o_2 \ldots o_k) \cdot \vec{M} &= (\vec{p} \cdot \vec{D}_0 \cdot \vec{M} \cdot \vec{D}_{o_2} \cdot \vec{M} \cdot \ldots \cdot \vec{D}_{o_k}) \cdot \vec{M} \\
&= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}\ , \qquad (3.8a) \\
\vec{P}_U(1 o_2 \ldots o_k) \cdot \vec{M} &= (\vec{p} \cdot \vec{D}_1 \cdot \vec{M} \cdot \vec{D}_{o_2} \cdot \vec{M} \cdot \ldots \cdot \vec{D}_{o_k}) \cdot \vec{M} \\
&= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}\ , \qquad (3.8b)
\end{aligned}$$

$$\vec{P}_U(@o_2 \dots o_k) \cdot \vec{M} = (\vec{p} \cdot \vec{D}_@ \cdot \vec{M} \cdot \vec{D}_{o_2} \cdot \vec{M} \cdot \ldots \cdot \vec{D}_{o_k}) \cdot \vec{M}$$
$$= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \end{bmatrix} \quad . \tag{3.8c}$$

The consistency proof will then be split into the three cases corresponding to $(o_1 = 0)$, $(o_1 = 1)$, and $(o_1 = @)$.

Case $(o_1 = 0)$*:* Replacing by (3.8a) and (3.6) in the state distribution computation for the strings x_i0, x_i1, and $x_i@$:

$$\vec{P}_U(x_i0) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_0$$
$$= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \cdot \begin{bmatrix} \vec{I}_n & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$
$$= \frac{1}{3^k} \left[\!\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1} \right]\!\!\right] \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$
$$= \frac{1}{3^{k+1}} \left[\!\!\left[\vec{e_1}, \vec{e_1}, \vec{e_1} \right]\!\!\right] \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$
$$= \left[\!\!\left[\frac{\vec{e_1} \cdot \vec{M}_{x_i}}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] = \boxed{\left[\!\!\left[\frac{\vec{e_i}}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right]} \quad .$$

$$\vec{P}_U(x_i1) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_1$$
$$= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{I}_n & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \frac{1}{3^k}\left[\!\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}\right]\!\!\right] \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{M}_{x_i} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \frac{1}{3^{k+1}}\left[\!\!\left[\vec{e_1}, \vec{e_1}, \vec{e_1}\right]\!\!\right] \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{M}_{x_i} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \left[\!\!\left[\vec{0}_n, \frac{\vec{e_1} \cdot \vec{M}_{x_i}}{3^{k+1}}, \vec{0}_n\right]\!\!\right] = \boxed{\left[\!\!\left[\vec{0}_n, \frac{\vec{e_i}}{3^{k+1}}, \vec{0}_n\right]\!\!\right]} \ .$$

$$\vec{P}_U(x_i@) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_@$$

$$= \frac{\vec{p}}{3^k} \cdot \begin{bmatrix} \vec{M}_{x_i} & \vec{M}_{x_i} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n \end{bmatrix}$$

$$= \frac{1}{3^k}\left[\!\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}\right]\!\!\right] \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \frac{1}{3^{k+1}}\left[\!\!\left[\vec{e_1}, \vec{e_1}, \vec{e_1}\right]\!\!\right] \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{M}_{x_i} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_1} \cdot \vec{M}_{x_i}}{3^{k+1}})\right]\!\!\right] = \boxed{\left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i}}{3^{k+1}}\right]\!\!\right]} \ .$$

The HMM U is therefore consistent with the transactions in T^h corresponding to strings whose first symbol is $(o_1 = 0)$.

The cases $(o_1 = 1)$, and $(o_1 = @)$ proceed similarly by using equations (3.8b) and (3.8c) instead of (3.8a), respectively, in the state distribution computations of the strings x_i0, x_i1, and $x_i@$. The HMM U is therefore consistent with the dataset T^h. □

While Theorem 3.3 does not depend on the structure of the original DFA, a similarly general result for the reciprocal proposition is not available, namely that if a HMM is consistent with T^h, then there is a DFA consistent with T. However, for the specific DFA dataset arising from a SAT' problem of Corollary 3.1, the reciprocal holds as shown in the following theorem.

Theorem 3.4. *Let C be a set of clauses in a* SAT' *problem. Let $T^{@}$ be the set of* DFA *examples obtained by applying the transformation described in Definition 3.5 to the set T of transitions associated with the* SAT' *problem for C described in Definition 3.3. Let T^h be the* HMM *example dataset as defined in Definition 3.7. Then if there is an* HMM *consistent with T^h, there exist a truth assignment satisfying all clauses of C from the* SAT'*problem.*

Let $U = (Q^h, \Sigma^{@}, \delta^h, \beta, \rho)$ be the HMM consistent with the set T^h. Let $\vec{p}$ be the stochastic row vector associated with ρ, and $\vec{D}_0$, $\vec{D}_1$, $\vec{D}_{@}$ be the diagonal $(3n \times 3n)$-dimensional matrices associated with the display probability distributions for the symbols 0, 1, and @, respectively (as described in Theorem 3.3).

Let

$$\vec{M} = \begin{bmatrix} \vec{A}_1 & \vec{A}_2 & \vec{A}_3 \\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3 \\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3 \end{bmatrix} \tag{3.9}$$

be the stochastic $(3n \times 3n)$-dimensional matrix associated with the TPD δ^h. $\vec{A}_1$, $\vec{A}_2$, $\vec{A}_3$, $\vec{B}_1$, $\vec{B}_2$, $\vec{B}_3$, $\vec{C}_1$, $\vec{C}_2$, $\vec{C}_3$ are all $(n \times n)$-dimensional matrices.

Theorem 3.4 will be proven by way of Lemmas 3.1, 3.2, 3.3, and 3.4.

Lemma 3.1. *The* ISPD *vector $\vec{p}$ for the* HMM *U from Theorem 3.4 is:*

$$\vec{p} = \left[\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1} \right]\!\right] \ .$$

Proof. From observing $T^{@}$, it is straightforward to determine the initial state of a DFA consistent with $T^{@}$, namely q_1. Notice that to be consistent with $T^{@}$, a DFA must satisfy $\delta(q_1, \lambda) = q_1$ (or equivalently $\langle\, \lambda\,, \vec{e_1}\, \rangle$), where λ represents the empty string ($|\lambda| = 0$), and therefore U must be consistent with the examples:

$$\left\langle\, 0\,, \left[\!\!\left[\frac{1}{3}\vec{e_1}, \vec{0}_n, \vec{0}_n, \right]\!\!\right] \right\rangle, \tag{3.10a}$$

$$\left\langle\, 1\,, \left[\!\!\left[\vec{0}_n, \frac{1}{3}\vec{e_1}, \vec{0}_n \right]\!\!\right] \right\rangle, \tag{3.10b}$$

$$\left\langle\, @\,, \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{1}{3}\vec{e_i} \right]\!\!\right] \right\rangle \tag{3.10c}$$

obtained from $\langle\, \lambda\,, \vec{e_1}\, \rangle$.

Therefore from (3.10a), (3.10b), and (3.10c):

$$\vec{P}_U(0) = \vec{p} \cdot \vec{D}_0 \;=\; \left[\!\!\left[\frac{1}{3}\vec{e_1}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \quad , \tag{3.11a}$$

$$\vec{P}_U(1) = \vec{p} \cdot \vec{D}_1 \;=\; \left[\!\!\left[\vec{0}_n, \frac{1}{3}\vec{e_1}, \vec{0}_n \right]\!\!\right] \quad , \tag{3.11b}$$

$$\vec{P}_U(@) = \vec{p} \cdot \vec{D}_@ = \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{1}{3}\vec{e_1} \right]\!\!\right] \quad . \tag{3.11c}$$

Summing up (3.11a), (3.11b), and (3.11c) and using (3.5):

$$\vec{p} \cdot (\vec{D}_0 + \vec{D}_1 + \vec{D}_@) = \vec{p} \cdot \vec{I}_{3n} = \left[\!\!\left[\frac{1}{3}\vec{e_i}, \frac{1}{3}\vec{e_i}, \frac{1}{3}\vec{e_i} \right]\!\!\right] \quad .$$

Hence, $\vec{p} = \left[\!\!\left[\frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1}, \frac{1}{3}\vec{e_1} \right]\!\!\right]$. □

Lemma 3.2. *The diagonal $(3n \times 3n)$-dimensional matrices associated with the display probability distributions of the* HMM *U from Theorem 3.4 are $\vec{D}_0$, $\vec{D}_1$, and $\vec{D}_@$ from* (3.4a), (3.4b), *and* (3.4c), *respectively.*

Proof. Let

$$\vec{D}_0 = \begin{bmatrix} \vec{J} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{K} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{L} \end{bmatrix}$$

where $\vec{J}$, $\vec{K}$, and $\vec{L}$ are three $(n \times n)$ diagonal matrices.

Since, per the dataset T construction, there exist at least an example $\langle\, x_i\,, q_i\,\rangle \in T$, for each state $q_i,\ i = 1, \ldots, n$ (every $q_i \in Q$ is reachable from at least one string x_i in T). Therefore, since $T \subseteq T^{@}$, from (3.2a), (3.2b), (3.2c), in Definition 3.7, it follows that:

For $i = 1, \ldots, n$:

$$\left\langle\, x_i 0\,,\, \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \,\right\rangle \in T^h \ , \tag{3.12a}$$

$$\left\langle\, x_i 1\,,\, \left[\!\!\left[\vec{0}_n, \frac{\vec{e_i}}{3^{|x_i|+1}}, \vec{0}_n \right]\!\!\right] \,\right\rangle \in T^h \ , \tag{3.12b}$$

$$\left\langle\, x_i @\,,\, \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i}}{3^{|x_i|+1}} \right]\!\!\right] \,\right\rangle \in T^h \ . \tag{3.12c}$$

Since U is consistent with T^h, from (3.12a), (3.12b), and (3.12c), respectively:

$$\vec{P}_U(x_i 0) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_0 \ = \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \ , \tag{3.13a}$$

$$\vec{P}_U(x_i 1) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_1 \ = \left[\!\!\left[\vec{0}_n, \frac{\vec{e_i}}{3^{|x_i|+1}}, \vec{0}_n \right]\!\!\right] \ , \tag{3.13b}$$

$$\vec{P}_U(x_i @) = \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_@ = \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i}}{3^{|x_i|+1}} \right]\!\!\right] \ . \tag{3.13c}$$

Since:

$$\begin{aligned} \vec{P}_U(x_i) \cdot \vec{M} \cdot (\vec{D}_0 + \vec{D}_1 + \vec{D}_@) &= \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{I}_{3n} \\ &= \vec{P}_U(x_i) \cdot \vec{M} \ , \end{aligned}$$

summing up (3.13a), (3.13b), and (3.13c) it follows that:

$$\vec{P}_U(x_i) \cdot \vec{M} = \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}} \right]\!\!\right] \ . \tag{3.14}$$

Thus, replacing from (3.14), for all $i = 1, \ldots, n$:

$$\begin{aligned}
\vec{P}_U(x_i 0) &= \vec{P}_U(x_i) \cdot \vec{M} \cdot \vec{D}_0 \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}} \right]\!\!\right] \cdot \vec{D}_0 \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}}, \frac{\vec{e_i}}{3^{|x_i|+1}} \right]\!\!\right] \cdot \begin{bmatrix} \vec{J} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{K} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{L} \end{bmatrix} \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{J}, \frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{K}, \frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{L} \right]\!\!\right] \quad . \qquad (3.15)
\end{aligned}$$

Then, from (3.12a) and (3.15):

$$\frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{J} = \frac{\vec{e_i}}{3^{|x_i|+1}} \ , \quad \frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{K} = \vec{0}_{n\times n} \ , \quad \frac{\vec{e_i}}{3^{|x_i|+1}} \cdot \vec{L} = \vec{0}_{n\times n} \ .$$

Therefore, for $i = 1, \ldots, n$:

$$\vec{J}[i,i] = 1, \quad \vec{K}[i,i] = 0, \text{ and } \quad \vec{L}[i,i] = 0 \ .$$

Or, equivalently:

$$\vec{J} = \vec{I}_n, \quad \vec{K} = \vec{0}_{n\times n}, \text{ and } \quad \vec{L} = \vec{0}_{n\times n} \ .$$

And therefore:

$$\vec{D}_0 = \begin{bmatrix} \vec{I}_n & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \ .$$

Similarly from (3.13b), and (3.13c) it respectively follows:

$$\vec{D}_1 = \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{I}_n & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}, \quad \text{and} \quad \vec{D}_@ = \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n \end{bmatrix} \quad . \qquad \square$$

Lemma 3.3. *Let* $\langle\, x_i \,,\, q_i \,\rangle$*, and* $\langle\, x_i\sigma \,,\, q_j \,\rangle$ *be two examples from the dataset* $T^{@}$ *such that* $\sigma \in \Sigma^{@}$*,* $x_i \in \Sigma^{+}$*, and* $\{q_i, q_j\} \subseteq Q$*. Let* $\vec{M}$ *be the stochastic* $(3n \times 3n)$*-dimensional matrix of* (3.9) *associated with the* TPD *for* U*:*

1. *If* $(\sigma = 0)$ *then* $\vec{e_i} \cdot \vec{A}_1 = \vec{e_i} \cdot \vec{A}_2 = \vec{e_i} \cdot \vec{A}_3 = \frac{1}{3}\vec{e_j}$ *.*
2. *If* $(\sigma = 1)$ *then* $\vec{e_i} \cdot \vec{B}_1 = \vec{e_i} \cdot \vec{B}_2 = \vec{e_i} \cdot \vec{B}_3 = \frac{1}{3}\vec{e_j}$ *.*
3. *If* $(\sigma = @)$ *then* $\vec{e_i} \cdot \vec{C}_1 = \vec{e_i} \cdot \vec{C}_2 = \vec{e_i} \cdot \vec{C}_3 = \frac{1}{3}\vec{e_j}$ *.*

Proof. Let $k = |x_i|$. Since $\langle\, x_i \,,\, q_i \,\rangle \in T^{@}$ and U is consistent with T^h, as argued in Lemma 3.2, (3.13a), (3.13b), and (3.13c), hold for the HMM U.
Additionally, since $\langle\, x_i\sigma \,,\, q_j \,\rangle \in T^{@}$:

$$\left\langle\, x_i\sigma 0 \,,\, \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \,\right\rangle \in T^h, \tag{3.16a}$$

$$\left\langle\, x_i\sigma 1 \,,\, \left[\!\!\left[\vec{0}_n, \frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n \right]\!\!\right] \,\right\rangle \in T^h, \tag{3.16b}$$

$$\left\langle\, x_i\sigma @ \,,\, \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_j}}{3^{k+2}} \right]\!\!\right] \,\right\rangle \in T^h, \tag{3.16c}$$

and therefore due to U being consistent with T^h:

$$\vec{P}_U(x_i\sigma 0) = \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \tag{3.17a}$$

$$\vec{P}_U(x_i\sigma 1) = \left[\!\!\left[\vec{0}_n, \frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n \right]\!\!\right] \tag{3.17b}$$

$$\vec{P}_U(x_i\sigma @) = \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \tag{3.17c}$$

The cases $(\sigma = 0)$, $(\sigma = 1)$, and $(\sigma = @)$ are handled separately.
Case $(\sigma = 0)$*:* by replacing $(\sigma = 0)$ in (3.17a), (3.17b), and (3.17c) the following is obtained:

$$\vec{P}_U(x_i 00) = \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \tag{3.18a}$$

$$\vec{P}_U(x_i01) = \left[\!\!\left[\vec{0}_n, \frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n \right]\!\!\right] \tag{3.18b}$$

$$\vec{P}_U(x_i0@) = \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \tag{3.18c}$$

On the other hand, from using (3.13a) in the state distribution computation for $\vec{P}_U(x_i00)$, $\vec{P}_U(x_i01)$, and $\vec{P}_U(x_i0@)$:

$$\begin{aligned}
\vec{P}_U(x_i00) &= \vec{P}_U(x0) \cdot M \cdot \vec{D}_0 \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{A}_2 & \vec{A}_3 \\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3 \\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3 \end{bmatrix} \cdot \begin{bmatrix} \vec{I}_n & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \\
&= \left[\!\!\left[\frac{\vec{e_i} \cdot \vec{A}_1}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right]
\end{aligned} \tag{3.19a}$$

$$\begin{aligned}
\vec{P}_U(x_i01) &= \vec{P}_U(x0) \cdot M \cdot \vec{D}_1 \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{A}_2 & \vec{A}_3 \\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3 \\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3 \end{bmatrix} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{I}_n & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix} \\
&= \left[\!\!\left[\vec{0}_n, \frac{\vec{e_i} \cdot \vec{A}_2}{3^{k+1}}, \vec{0}_n \right]\!\!\right]
\end{aligned} \tag{3.19b}$$

$$\begin{aligned}
\vec{P}_U(x_i0@) &= \vec{P}_U(x0) \cdot M \cdot \vec{D}_@ \\
&= \left[\!\!\left[\frac{\vec{e_i}}{3^{k+1}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{A}_2 & \vec{A}_3 \\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3 \\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3 \end{bmatrix} \cdot \begin{bmatrix} \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n \end{bmatrix} \\
&= \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i} \cdot \vec{A}_3}{3^{k+1}} \right]\!\!\right]
\end{aligned} \tag{3.19c}$$

From (3.19a) and (3.18a):

$$\begin{aligned}
\vec{P}_U(x_i00) &= \left[\!\!\left[\frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_1, \vec{0}_n, \vec{0}_n \right]\!\!\right] = \left[\!\!\left[\frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \\
\frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_1 &= \frac{\vec{e_j}}{3^{k+2}} \\
\vec{e_i} \cdot \vec{A}_1 &= \frac{\vec{e_j}}{3} \ . \qquad (3.20a)
\end{aligned}$$

From (3.19b) and (3.18b):

$$\begin{aligned}
\vec{P}_U(x_i01) &= \left[\!\!\left[\vec{0}_n, \frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_2, \vec{0}_n \right]\!\!\right] = \left[\!\!\left[\vec{0}_n, \frac{\vec{e_j}}{3^{k+2}}, \vec{0}_n \right]\!\!\right] \\
\frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_2 &= \frac{\vec{e_j}}{3^{k+2}} \\
\vec{e_i} \cdot \vec{A}_2 &= \frac{\vec{e_j}}{3} \ . \qquad (3.20b)
\end{aligned}$$

From (3.19c) and (3.18c):

$$\begin{aligned}
\vec{P}_U(x_i0@) &= \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_3, \right]\!\!\right] = \left[\!\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_j}}{3^{k+2}} \right]\!\!\right] \\
\frac{\vec{e_i}}{3^{k+1}} \cdot \vec{A}_3 &= \frac{\vec{e_j}}{3^{k+2}} \\
\vec{e_i} \cdot \vec{A}_3 &= \frac{\vec{e_j}}{3} \ . \qquad (3.20c)
\end{aligned}$$

Finally, from (3.20a), (3.20b), and (3.20c):

$$\boxed{\vec{e_i} \cdot \vec{A}_1 = \vec{e_i} \cdot \vec{A}_2 = \vec{e_i} \cdot \vec{A}_3 = \frac{1}{3}\vec{e_j}} \ .$$

The remaining cases proceed in a similar manner from (3.13b) together with (3.17a), (3.17b), and (3.17c) for the case ($\sigma = 1$); and (3.13c), (3.17a), (3.17b), and (3.17c) for the case ($\sigma = @$) case. The details can be easily verified and are omitted here. □

Corollary 3.2. $\vec{B}_1 = \vec{B}_2 = \vec{B}_3$.

Proof. In the dataset T for a DFA from the SAT$'$ problem, the transition $\delta(q_i, 1)$ is defined for every $q_i \in Q$, and since every state q_i is reachable by at least one string x_i in the dataset, it can be concluded that for every $\langle\, x_i\,, q_i\,\rangle \in T$, there exists an example $\langle\, x_i 1\,, q_j\,\rangle \in T$. Since $T \subseteq T^{@}$, it follows that for every $q_i \in Q$, $\langle\, x_i\,, q_i\,\rangle \in T^{@}$, and $\langle\, x_i 1\,, q_j\,\rangle \in T^{@}$. Consequently, Lemma 3.3 (case $\sigma = 1$), applies for all $q_i \in Q$.

Namely:

$$\vec{e_i} \cdot \vec{B}_1 = \vec{e_i} \cdot \vec{B}_2 = \vec{e_i} \cdot \vec{B}_3 \qquad \forall(\, 1 \leq i \leq n) \ .$$

Equivalently, since $[\vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T] = \vec{I}_n$:

$$[\vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T] \cdot \vec{B}_1 = [\vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T] \cdot \vec{B}_2 = [\vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T] \cdot \vec{B}_3 \ ,$$
$$\vec{I}_n \cdot \vec{B}_1 = \vec{I}_n \cdot \vec{B}_2 = \vec{I}_n \cdot \vec{B}_3 \ .$$

And therefore:

$$\boxed{\vec{B}_1 = \vec{B}_2 = \vec{B}_3} \ . \qquad \square$$

Corollary 3.3. $\vec{C}_1 = \vec{C}_2 = \vec{C}_3 = \frac{1}{3}\vec{I}_n$.

Proof. Since for each example $\langle\, x_i\,, q_i\,\rangle \in T$, the example $\langle\, x_i @\,, q_i\,\rangle \in T^{@}$, and $T \subseteq T^{@}$, it follows that for every $q_i \in Q$, $\langle\, x_i\,, q_i\,\rangle \in T^{@}$, and $\langle\, x_i @\,, q_i\,\rangle \in T^{@}$. Therefore, Lemma 3.3 (case $\sigma = @$), applies for all $q_i \in Q$.

It follows that:

$$\vec{e_i} \cdot \vec{C}_1 = \vec{e_i} \cdot \vec{C}_2 = \vec{e_i} \cdot \vec{C}_3 = \frac{1}{3}\vec{e_i} \qquad \forall(\, 1 \leq i \leq n) \ .$$

Since $[\, \vec{e_1}^T, \vec{e_2}^T, \ldots, \vec{e_n}^T\,] = \vec{I}_n$:

$$\vec{I}_n \cdot \vec{C}_1 = \vec{I}_n \cdot \vec{C}_2 = \vec{I}_n \cdot \vec{C}_3 = \frac{1}{3}\vec{I}_n \ .$$

Therefore:

$$\boxed{\vec{C}_1 = \vec{C}_2 = \vec{C}_3 = \frac{1}{3}\vec{I}_n} \ . \qquad \square$$

Corollary 3.4. *In constructing the* HMM U *consistent with* T^h*, the only rows of the matrix* $\vec{M}$ *from* (3.9) *that remain to be determined, are those in the* $(n \times n)$ *matrices* $\vec{A}_1, \vec{A}_2$*, and* $\vec{A}_3$*, which correspond to the leaf nodes of the trees* $q_{c_0}(\{q_{c_0,j} : j = 1, \ldots, r'\})$*, and the leaf nodes of the tree* $q_v(\{q_{v_i} : i = 1, \ldots, l'\})$*.*

Proof. It follows directly from the DFA construction as described in Sec. 3. □

Lemma 3.4. *Let* $c_j, (1 \leq j \leq r)$ *be a clause in the original* SAT' *problem. Let* h *be the index of the state* $q_{c_0,j}$ *corresponding to the input string* $1x_{c_j}$ *—i.e.* $\left\langle 1x_{c_j}, q_h \right\rangle = \left\langle 1x_{c_j}, q_{c_0,j} \right\rangle \in T$*. Let* $\vec{A}_1[h]$*,* $\vec{A}_2[h]$*,* $\vec{A}_2[h]$ *be the* h^{th} *row of* $\vec{A}_1$*,* $\vec{A}_2$*, and* $\vec{A}_3$ *respectively, in the matrix* $\vec{M}$ *of* (3.9) *fitting* T^h*. Then: if* $\vec{A}_3[h,t] > 0$ *for some* $1 \leq t \leq n$ *it must be the case that the state* q_t *corresponds to some leaf node* q_{v_i} *of the tree* q_v *in the DFA. Moreover, it is the case that* $v_i \in c_j$ *.*

Proof. Assuming, without loss of generality, c_j to be a positive clause and $|x_{c_j}| = k$, consider the training examples in T^h arising from the examples corresponding to the DFA transitions defined by T_4:

$$\left\langle 1x_{c_j}0@00 \ , \left[\!\left[\frac{\vec{e_4}}{3^{k+5}}, \vec{0}_n, \vec{0}_n \right]\!\right] \right\rangle \ , \tag{3.21a}$$

$$\left\langle 1x_{c_j}0@01 \ , \left[\!\left[\vec{0}_n, \frac{\vec{e_4}}{3^{k+5}}, \vec{0}_n \right]\!\right] \right\rangle \ , \tag{3.21b}$$

$$\left\langle 1x_{c_j}0@0@ \ , \left[\!\left[\vec{0}_n, \vec{0}_n, \frac{\vec{e_4}}{3^{k+5}} \right]\!\right] \right\rangle \ . \tag{3.21c}$$

Since U is consistent with T^h:

$$\vec{P}_U(1x_{c_j}0@) = \vec{P}_U(1x_{c_j}0) \cdot \vec{M} \cdot \vec{D}_{@} \ .$$

Additionally:

$$\vec{P}_U(1x_{c_j}0) = \left[\!\left[\frac{\vec{e_h}}{3^{k+2}}, \vec{0}_n, \vec{0}_n \right]\!\right] \ .$$

Therefore:

$$\begin{aligned}
\vec{P}_U(1x_{c_j}0@) &= \vec{P}_U(1x_{c_j}0)\cdot\vec{M}\cdot\vec{D}_@ \\
&= \left[\!\!\left[\frac{\vec{e_h}}{3^{k+2}},\vec{0}_n,\vec{0}_n\right]\!\!\right]\cdot\vec{M}\cdot\vec{D}_@ \\
&= \left[\!\!\left[\frac{\vec{e_h}}{3^{k+2}},\vec{0}_n,\vec{0}_n\right]\!\!\right]\cdot\begin{bmatrix}\vec{A}_1 & \vec{A}_2 & \vec{A}_3\\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3\\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3\end{bmatrix}\cdot\begin{bmatrix}\vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n\end{bmatrix} \\
&= \left[\!\!\left[\frac{\vec{e_h}\cdot\vec{A}_1}{3^{k+2}},\frac{\vec{e_h}\cdot\vec{A}_2}{3^{k+2}},\frac{\vec{e_h}\cdot\vec{A}_3}{3^{k+2}}\right]\!\!\right]\cdot\begin{bmatrix}\vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{I}_n\end{bmatrix} \\
&= \left[\!\!\left[\vec{0}_n,\vec{0}_n,\frac{\vec{e_h}\cdot\vec{A}_3}{3^{k+2}}\right]\!\!\right] = \left[\!\!\left[\vec{0}_n,\vec{0}_n,\frac{\vec{A}_3[h]}{3^{k+2}}\right]\!\!\right] \quad .
\end{aligned}$$

Similarly:

$$\begin{aligned}
\vec{P}_U(1x_{c_j}0@00) &= \vec{P}_U(1x_{c_j}0@)\cdot\vec{M}\cdot\vec{D}_0\cdot\vec{M}\cdot\vec{D}_0 \\
&= \left[\!\!\left[\vec{0}_n,\vec{0}_n,\frac{\vec{e_h}\cdot\vec{A}_3}{3^{k+2}}\right]\!\!\right]\cdot\left(\vec{M}\cdot\vec{D}_0\right)^2 \\
&= \left[\!\!\left[\vec{0}_n,\vec{0}_n,\frac{\vec{A}_3[h]}{3^{k+2}}\right]\!\!\right]\cdot\left(\begin{bmatrix}\vec{A}_1 & \vec{A}_2 & \vec{A}_3\\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3\\ \vec{C}_1 & \vec{C}_2 & \vec{C}_3\end{bmatrix}\cdot\begin{bmatrix}\vec{I}_n & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{0}_{n\times n} & \vec{0}_{n\times n} & \vec{0}_{n\times n}\end{bmatrix}\right)^2 \\
&= \left[\!\!\left[\vec{0}_n,\vec{0}_n,\frac{\vec{A}_3[h]}{3^{k+2}}\right]\!\!\right]\cdot\begin{bmatrix}\vec{A}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{B}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{C}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\end{bmatrix}\cdot\begin{bmatrix}\vec{A}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{B}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\\ \vec{C}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n}\end{bmatrix}
\end{aligned}$$

$$= \left[\!\!\left[\frac{\vec{A}_3[h]}{3^{k+2}} \cdot \vec{C}_1, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{B}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{C}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \left[\!\!\left[\frac{\vec{A}_3[h]}{3^{k+2}} \cdot \frac{1}{3}\vec{I}_n, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{B}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{C}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \left[\!\!\left[\frac{\vec{A}_3[h]}{3^{k+3}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \cdot \begin{bmatrix} \vec{A}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{B}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \\ \vec{C}_1 & \vec{0}_{n\times n} & \vec{0}_{n\times n} \end{bmatrix}$$

$$= \left[\!\!\left[\frac{\vec{A}_3[h] \cdot \vec{A}_1}{3^{k+3}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \quad .$$

On the other hand, from (3.21a):

$$\vec{P}_U(1x_{c_j}0@00) = \left[\!\!\left[\frac{\vec{e_4}}{3^{k+5}}, \vec{0}_n, \vec{0}_n \right]\!\!\right] \quad ,$$

which implies that:

$$\frac{\vec{e_4}}{3^{k+5}} = \frac{\vec{A}_3[h]}{3^{k+3}} \cdot \vec{A}_1$$

Or equivalently:

$$\vec{A}_3[h] \cdot \vec{A}_1 = \frac{1}{3^2}\vec{e_4} \ .$$

Similar derivations involving (3.21b) and (3.21c) produce:

$$\vec{A}_3[h] \cdot \vec{A}_2 = \vec{A}_3[h] \cdot \vec{A}_3 = \frac{1}{3^2}\vec{e_4} \ .$$

It follows that if $\vec{A}_3[h,t] > 0$, the t^{th} row of $\vec{A}_1, \vec{A}_2$, and $\vec{A}_3$ $(\vec{A}_1[t], \vec{A}_2[t], \vec{A}_3[t])$, must be of form $(\alpha \times \vec{e_4}), (\alpha \geq 0)$.

Namely:

$$\forall (1 \leq i \leq n): \text{ if } i \neq 4 \text{ then } \vec{A}_1[t,i] = \vec{A}_2[t,i] = \vec{A}_3[t,i] = 0 \ .$$

From Corollary 3.4 of Lemma 3.3, the only rows yet to be determined in constructing the HMM U consistent with T^h are those in $\vec{A}_1, \vec{A}_2$, and $\vec{A}_3$ that correspond to states $q_{c_0,p}$ for some clause $c_p (1 \leq p \leq r)$, or to states q_{v_i} for some literal $v_i (1 \leq i \leq l)$, since the rows of $\vec{A}_1, \vec{A}_2$, and $\vec{A}_3$ are already defined, none is of the form $(\alpha \times \vec{e_4})$.

However, it will be shown next that if $\vec{A}_3[h,t] > 0$, state q_t must be equal to q_{v_i} for some v_i, $(1 \leq i \leq n)$, it can not be the case that $q_t = q_{c_0,p}$ for clause c_p, $(1 \leq \ p \leq r)$.

Assume that $\vec{A}_3[h,t] > 0$, $q_t = q_{c_0,p}$ for some $c_p, (1 \leq p \leq r)$.

From the DFA transitions defined by T_4, and assuming c_p to be a positive clause and $|x_p| = k$, it follows that $\left\langle 1x_{c_p}000\,, \left[\!\left[\frac{1}{3^{k+5}}\vec{e_4}, \vec{0}_n, \vec{0}_n \right]\!\right] \right\rangle \in T^h$. Using similar arguments, it can be shown that for the t^{th} row of $\vec{A}_1$, namely $\vec{A}_1[t]$, if $\vec{A}_1[t,i] > 0$ then the i^{th} row of $\vec{A}_1, \vec{A}_2, \vec{A}_3$ must be of the form $(\alpha \times \vec{e_4}), (\alpha \geq 0)$. This, however, gives a contradiction. Because from $\vec{A}_3[h,t] > 0$, it has been shown that $\vec{A}_1[t]$ is of the form $(\alpha \times \vec{e_4}), (\alpha \geq 0)$. Thus $\vec{A}_1[t,4] > 0$, which implies that $\vec{A}_1[4]$ is of the form $(\alpha \times \vec{e_4})$. However this is not possible since $\vec{A}_1[4] = \vec{A}_2[4] = \vec{A}_3[4] = \frac{1}{3}\vec{e_6}$.

Moreover, because of the training data:

$$\left\langle 1x_{c_j}0@1x_{c_j}00\,, \left[\!\left[\frac{\vec{e_2}}{3^{2k+6}}, \vec{0}_n, \vec{0}_n \right]\!\right] \right\rangle, \text{ and either:}$$
$$\left\langle 0x_{v_i}1x_{c_j}00 \quad , \left[\!\left[\frac{\vec{e_2}}{3^{2k+6}}, \vec{0}_n, \vec{0}_n \right]\!\right] \right\rangle \text{ if } v_i \in c_j, \text{ or}$$
$$\left\langle 0x_{v_i}1x_{c_j}00 \quad , \left[\!\left[\frac{\vec{e_3}}{3^{2k+6}}, \vec{0}_n, \vec{0}_n \right]\!\right] \right\rangle \text{ if } v_i \notin c_j \ ,$$

it follows that if $\vec{A}_3[h,t] > 0$ then $q_t = q_{v_i},\ v_i \in c_j$. □

Lemma 3.4 guarantees that if there is an HMM consistent with T^h, a truth assignment can be constructed as follows:

For each clause c_j such that $q_{c_0,j} = q_h$:

– if $\vec{A}_3[h,t] > 0$, then $q_t = q_{v_i}$.

$$\text{Assign } v_i = \begin{cases} \text{true} , & \text{if } c_j \text{ is positive} , \\ \text{false} , & \text{if } c_j \text{ is negative} \end{cases}$$

– Assign all remaining v_j arbitrarily.

In order to show the truth assignment to be valid, suppose $c_{j_1} \in C$, and $c_{j_2} \in C$ are two clauses one being positive (consisting only of positive literals) and the other being negative. Let h_1 and h_2 be the rows of the matrix $\vec{A}_3$ corresponding to clauses c_{j_1} and c_{j_2}, respectively. Then as shown in Lemma 3.4, it cannot be the case that $\vec{A}_3[h_1, t] > 0$ and $\vec{A}_3[h_2, t] > 0$ some column $t \in \{1, \ldots, n\}$. Assuming c_{j_1} is the positive clause the row t of $\vec{A}_3$ is of the form $\vec{A}_3[t] = \alpha \vec{e_4}$ for some $\alpha \geq 0$, while c_{j_2} being a negative clause implies that row $\vec{A}_3[t] = \alpha \vec{e_5}$ for some $\alpha \geq 0$. Therefore only one of $\vec{A}(h_1, t)$ and $\vec{A}(h_2, t)$ can be strictly positive proving the truth assignment to be valid.

This concludes the proof of Theorem 3.4. Theorems 3.3, and 3.4 respectively prove the forward and backward directions of Theorem 3.1.

Chapter 4:
Unsupervised HMM Learning

As shown by Theorem 3.1, the learning algorithm's ability to request the state distribution vectors of arbitrary strings from the SD oracle is crucial for efficiently learning the parameters of a target HMM (assuming $\mathcal{P} \neq \mathcal{NP}$).

In order to solve the matrix systems involved in computing the TPD, DPD, and ISPD for a target HMM $U = (Q, \Sigma, \delta, \beta, \rho)$ where $|Q| = n$ and $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, the proposed supervised learning algorithm *SupLearnHMM* from Fig. 2.1, requires the availability of the state distribution vectors corresponding to the strings in the following sets:

- $\{\sigma_1, \ldots, \sigma_m\} = \Sigma$,
- $X = \{x_1, x_2, \ldots, x_n\} \subset \Sigma^+$ such that $\left\{\vec{P}_U(x_1), \vec{P}_U(x_2), \ldots, \vec{P}_U(x_n)\right\}$ forms a linearly independent set,
- $X' = \{x_1\sigma_1, x_2\sigma_1, \ldots, x_n\sigma_1\} \cup \{x_1\sigma_2, x_2\sigma_2, \ldots, x_n\sigma_2\} \cup \cdots \cup \{x_1\sigma_m, x_2\sigma_m, \ldots, x_n\sigma_m\}$. The set of $(n \times m)$ suffix strings obtained by concatenating every symbol in Σ to the strings in X.

The state distribution vectors of every symbol in Σ are needed to compute the HMM's ISPD. The set X of n strings with linearly independent state distribution vectors, as well as the state distribution vectors of all the one-symbol suffixes of the strings in X (the strings in X') are used in the computation of the HMM's TPD and DPD. It is then worth exploring under what conditions on the HMM and training dataset, the availability of such sets of strings —and associated distribution vectors— to the learning algorithm can be guaranteed in the absence of the SD oracle.

Note that the required set of strings $(\Sigma \cup X \cup X')$ has cardinality $|\Sigma \cup X \cup X'| \leq m + n + (n \times m)$ since the set X needs not be disjoint with either Σ, nor X'. The number

of strings —and associated state distribution vectors— needed by the learning algorithm is therefore polynomial in n and m.

In this chapter, an alternative unsupervised framework, Probably Approximately Correct (PAC) Learning under Helpful Distributions, is described, where the learning algorithm queries an alternative oracle instead of the SD oracle, namely the EX oracle. Under this setting, the learning algorithm relinquishes the ability to request the state distribution vectors of specific strings from the oracle, which can only provide a randomly drawn training set from the instance dataset of examples.

In order to compensate for the lack of an SD oracle and for learning to become feasible, the EX oracle is assumed to draw the training examples from the instance set according to a probability distribution from a family of helpful distributions. The examples consist of pairs of the form $\left\langle x\,, \vec{P}_U(x) \right\rangle$ where the length of the strings are restricted by an integer bound. Next section introduces the standard PAC-learning setting as well as

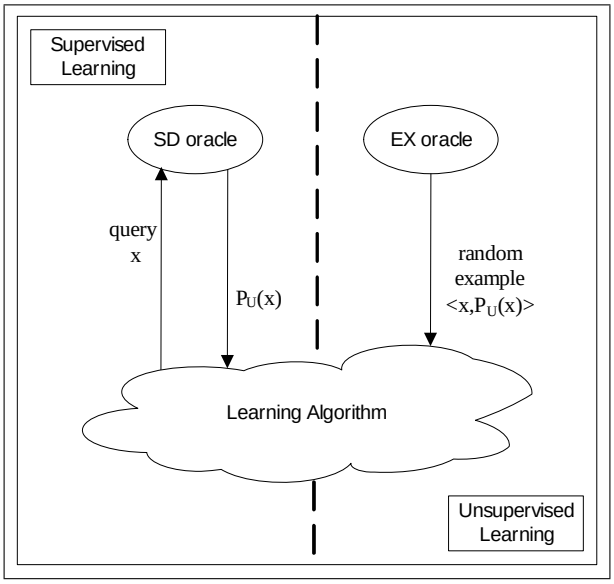

Figure 4.1: Learning under the supervised and unsupervised settings.

PAC-learning under helpful distributions. Section 4.2 defines a family of helpful probability distributions $\mathcal{D}_r$ on the instance set of examples for HMM learning. In Sec. 4.3 a

proposed unsupervised learning algorithm in the framework of PAC-learning under helpful distributions is given. The algorithm is shown to have polynomial complexity for the family of helpful distributions $\mathcal{D}_r$.

4.1 Probably Approximately Correct Learning

The Probably Approximately Correct learning framework is a well-known unsupervised learning model first introduced by Valiant (Valiant 1984). Under the PAC learning setting, a learning algorithm can request random finite length examples from an oracle EX, in order to (approximately) learn a target concept. The EX oracle draws the training examples from a set of instances following an unknown probability distribution. The framework relaxes the requirements of the learning algorithm with respect to the exact learning setting in two aspects:

- the learning algorithm is not required to learn the target concept exactly (with zero error) but may present an approximate hypothesis as long as the error in the approximation can be bounded by an arbitrarily small constant ϵ (precision).

- the learning algorithm is allowed to fail as long as the probability of failure can be bounded by an arbitrarily small constant θ (confidence). This takes into account that the algorithm may not be able to learn the target concept from every single set of random training examples.

More formally:

Definition 4.1. Let $\mathcal{C}$ be a concept class defined over a set of instances T. Let $\mathcal{D}$ be a probability distribution. An algorithm $\mathcal{A}$ is a *PAC learning algorithm* for $\mathcal{C}$ if $\mathcal{A}$ takes as input ϵ such that $0 < \epsilon < 1$, θ such that $0 < \theta < 1$, an integer k and for all concepts $c \in \mathcal{C}$ with $|c| \leq k$ and all arbitrary distributions $\mathcal{D}$, $\mathcal{A}$ is given access to oracle EX and $\mathcal{A}$ outputs $c' \in \mathcal{C}$, such that with probability at least $(1 - \theta)$, $error_{\mathcal{D}}(c') \leq \epsilon$.

The integer k is a bound on the size of the representation for the concepts in C. For the case of HMMs the representation size is $n^2 + n \times m - n - 1 = \mathcal{O}(n \times \max\{n, m\})$. The size comes from the fact that for an HMM with n states and m alphabet symbols, the ISPD can be represented by an n-dimensional stochastic vector ($n - 1$ degrees of freedom), the TPD can be represented by an $(n \times n)$ stochastic matrix ($n \times (n - 1)$ degrees of freedom), and the DPD can be represented by an $n \times m$ matrix ($n \times (m - 1)$ degrees of freedom). The total HMM representation size is obtained as the sum of the degrees of freedom for the ISPD, TPD, and DPD.

The $error_{\mathcal{D}}(c')$, or *true error* of the hypothesis c' (the learned concept) with respect to the target concept c and distribution $\mathcal{D}$ is the probability that c' will misclassify an instance drawn at random according to $\mathcal{D}$,

$$error_{\mathcal{D}}(c') = Pr_{\mathcal{D}}\left(\{x \in T : c(x) \neq c'(x)\}\right)$$

where the probability is taken over the instance distribution $\mathcal{D}$ (Mitchell 1997).

Definition 4.2. $\mathcal{C}$ is *PAC learnable* if there exists a PAC learning algorithm $\mathcal{A}$ for $\mathcal{C}$ which runs in time polynomial in $\epsilon^{-1}, \theta^{-1}$, and k.

The definitions given above prevent many concepts from being polynomially learnable under the PAC model mostly because of the requirement that the learning algorithm must produce an answer for any arbitrary distribution $\mathcal{D}$. In practice this restriction is not always reasonable since the instance set of examples is usually not arbitrary or random but rather conditioned by the target concept it represents. This has given rise to a number of variations of the model that assume $\mathcal{D}$ to be an unknown but helpful distribution (Denis and Gilleron 1997).

Definition 4.3. An algorithm $\mathcal{A}$ is a *PAC learning algorithm under helpful distributions* for a class $\mathcal{C}$ of concepts if $\mathcal{A}$ takes as input ϵ such that $0 < \epsilon < 1$, θ such that $0 < \theta < 1$, an integer k and for all concepts $c \in \mathcal{C}$ with $|c| \leq k$ and all helpful distributions $\mathcal{D}$, $\mathcal{A}$

is given access to oracle EX and $\mathcal{A}$ outputs $c' \in \mathcal{C}$, such that with probability at least $(1-\theta)$, $error_{\mathcal{D}}(c') \leq \epsilon$.

Definition 4.4. $\mathcal{C}$ is *PAC learnable under helpful distributions* $\mathcal{D}$ if there exists a PAC learning algorithm $\mathcal{A}$ for $\mathcal{C}$ under helpful distributions which runs in time polynomial in $\epsilon^{-1}, \theta^{-1}$, and k.

A family of distributions $\mathcal{D}_r$ are defined next. It will be shown that HMMs are PAC-learnable under these helpful distributions.

4.2 Helpful Distributions

Let $T_{\mathcal{D}}$ be an instance set of examples of the form $\left\langle x\,, \vec{P}_U(x) \right\rangle$ where x is a string of length $|x| \leq L, (L > 1)$ from an alphabet Σ of m symbols, and $\vec{P}_U(x)$ is its associated state distribution vector in the target HMM U.

The instance set $T_{\mathcal{D}}$ has an associated probability distribution $\mathcal{D}$ according to which the oracle EX draws the training examples from the instance dataset. The probability of an individual string (together with its state distribution vector) being drawn from the dataset $T_{\mathcal{D}}$ by the oracle is therefore given by $\mathcal{D}(x)$.

In order to define the family of helpful distributions, a number of auxiliary functions and lemmas will be introduced first.

Let $x = o_1 o_2 \ldots o_k$ be a k-length string from an alphabet $\Sigma, (o_i \in \Sigma, 1 \leq i \leq k)$. The alphabet Σ will be assumed without any loss of generality to have cardinality $|\Sigma| \geq 2$.

Let $\Psi : \Sigma^+ \rightarrow \mathbb{N}$ be a function, $\Psi(x) = \sum_{i=2}^{k} z_i$ such that:

$$z_i = \begin{cases} 0 & \text{if } o_i = o_{i-1} \ , \\ 1 & \text{if } o_i \neq o_{i-1} \ , \end{cases} \qquad (2 \leq i \leq k) \ .$$

The function $\Psi(x)$ counts the number of symbol changes occurring in a string x, in other words it gives the number of symbols that differ from the symbol in the preceding position of x. Table 4.1 shows several values of the function Ψ for a few example strings.

Table 4.1: Example values of $\Psi(x)$ for several strings x.

x	$\Psi(x)$
1	0
11111	0
11110	1
11011	2
1110001	2
1100101	4
10101011	6

Let L, c, and k be positive integers such that $0 \leq c < k \leq L$.

Let $X_{k,c} = \{x \in \Sigma^+ : |x| = k \text{ and } \Psi(x) = c\}$ be the set of k-length strings from the alphabet Σ that contain exactly c symbol changes ($\Psi(x) = c$).

Lemma 4.1. *Let c and k be integers such that $0 \leq c < k$, then the cardinality of the set $X_{k,c}$ is $|X_{k,c}| = m \times \binom{k-1}{c} \times (m-1)^c$.*

Proof. For any arbitrary string $x = o_1 \dots o_k$ of length k there are exactly $m = |\Sigma|$ different symbols in the alphabet to choose from for the first symbol o_1 of x. This leaves $(k-1)$ positions in the string x from which to choose a subset of c positions that will differ from the preceding symbol, giving a total of $\binom{k-1}{c}$ subset choices. Finally each of the c positions in the chosen subset can contain any of $(m-1)$ symbols (they cannot display the same symbol as the preceding position in the string).

Therefore $|X_{k,c}| = m \times \binom{k-1}{c} \times (m-1)^c$. □

Lemma 4.2. *Let c and L be integers such that $0 \leq c < L$ then:*

$$\sum_{k=c+1}^{L} \binom{k-1}{c} = \binom{L}{c+1} \tag{4.1}$$

Proof. The proof proceeds by mathematical induction on L.

– Base step: $L = 1$ implies $c = 0$ since $0 \leq c < L = 1$.

The left hand side of (4.1) is then:

$$\sum_{k=1}^{1} \binom{k-1}{0} = \binom{1-1}{0} = \binom{0}{0} = 1 \ .$$

Similarly the right hand side of (4.1) is

$$\binom{1}{0+1} = \binom{1}{1} = 1 \ .$$

– Inductive step: Assuming the (4.1) holds for strings of length up to L:

$$\sum_{k=c+1}^{L+1} \binom{k-1}{c} = \sum_{k=c+1}^{L} \binom{k-1}{c} + \binom{(L+1)-1}{c}$$

From the inductive hypothesis and using Pascal's Identity:

$$\sum_{k=c+1}^{L+1} \binom{k-1}{c} = \binom{L}{c+1} + \binom{L}{c} = \binom{L+1}{c+1} \ .$$

$$\therefore \quad \boxed{\sum_{k=c+1}^{L} \binom{k-1}{c} = \binom{L}{c+1}} \qquad \square$$

A new function $\Phi(L, c)$ is now defined that counts the number of strings x of length $|x| \leq L$ (lengths up to L) from an alphabet of m symbols such that each string contains exactly c symbol changes ($\Psi(x) = c$). Let $\mathcal{X}_{L,c} = \bigcup_{k=c+1}^{L} X_{k,c}$.

$$\Phi(L, c) = |\mathcal{X}_{L,c}| = \sum_{k=c+1}^{L} \left|\{x \in \Sigma^+ : |x| = k \text{ and } \Psi(x) = c\}\right| \ .$$

From Lemma 4.1:

$$= \sum_{k=c+1}^{L} m \times \binom{k-1}{c} \times (m-1)^c$$
$$= m \times (m-1)^c \times \sum_{k=c+1}^{L} \binom{k-1}{c}$$

And replacing from Lemma 4.2:

$$\boxed{\Phi(L,c) = m \times (m-1)^c \times \binom{L}{c+1}} \ . \tag{4.2}$$

The family of helpful probability distributions $\mathcal{D}_r$ can now be defined on the training dataset of examples containing strings of lengths up to L, where r is any constant, $r \geq 2$, and $c = \Psi(x)$:

$$\mathcal{D}_r(x) = \frac{(r-1)\ r^{(L-c-1)}}{\Phi(L,c)(r^L - 1)}$$

Replacing $\Phi(L,c)$ as per (4.2):

$$\boxed{\mathcal{D}_r(x) = \frac{(r-1)\ r^{(L-c-1)}}{m(m-1)^c\ \binom{L}{c+1}\ (r^L - 1)}} \ . \tag{4.3}$$

It is clear from (4.3) that under the conditions that $m > 1$, $r \geq 2$, and $0 \leq c < L$, $\mathcal{D}_r(x)$ is non-negative and well defined for all strings x of length up to L. It remains to be shown that $\mathcal{D}_r$ also satisfies the requirement that the sum of its image equals unity.

Theorem 4.1.

$$\sum_{c=0}^{L-1} \sum_{x \in \mathcal{X}_{L,c}} \mathcal{D}_r(x) = 1 \ .$$

Proof.

$$\sum_{c=0}^{L-1} \sum_{x \in \mathcal{X}_{L,c}} \mathcal{D}_r(x) = \sum_{c=0}^{L-1} \sum_{x \in \mathcal{X}_{L,c}} \frac{(r-1)\ r^{(L-c-1)}}{m(m-1)^c \binom{L}{c+1} (r^L - 1)}$$

Since each string $x \in \mathcal{X}_{L,c}$ has the same probability $\mathcal{D}_r(x)$ given by (4.3) and there are exactly $|\mathcal{X}_{L,c}| = \Phi(L,x)$ strings in $\mathcal{X}_{L,c}$:

$$\begin{aligned}
&= \sum_{c=0}^{L-1} \Phi(L,c) \times \frac{(r-1)\ r^{(L-c-1)}}{m(m-1)^c \binom{L}{c+1} (r^L - 1)} \\
&= \sum_{c=0}^{L-1} \frac{(r-1)\ r^{(L-c-1)}}{(r^L - 1)} \\
&= \frac{(r-1)\ r^{(L-1)}}{(r^L - 1)} \times \sum_{c=0}^{L-1} r^{-c} \\
&= \frac{(r-1)\ r^{(L-1)}}{(r^L - 1)} \times \frac{\left(r^{-L} - 1\right)}{(r^{-1} - 1)} \\
&= \frac{(r-1)\ r^{(L-1)}}{(r^L - 1)} \times \frac{\left(1 - r^L\right) r^{-(L-1)}}{(1-r)} \\
&= \frac{(r-1)\ r^{(L-1)}}{(r^L - 1)} \times \frac{\left(r^L - 1\right) r^{-(L-1)}}{(r-1)} \\
&= 1 \ . \qquad \square
\end{aligned}$$

4.3 HMM PAC Learning Under Helpful Distributions

To prove that HMMs are PAC-learnable under the family of helpful distributions $\mathcal{D}_r$, it needs to be shown that the sample complexity of the PAC-learning algorithm (the number of calls to the EX oracle) is polynomial in $\epsilon^{-1}, \theta^{-1}$, m, and n .

The PAC-learning algorithm needs to request a sufficient number of training examples from the EX oracle as to ensure with high confidence that the strings contained in the set $(\Sigma + X + X')$ are included in the training set (the probability of the examples

corresponding to the strings in $(\Sigma + X + X')$ not being present in the training example gathered from the oracle EX has to have θ for an upper bound).

In this setting, the EX oracle draws a set of training examples from an instance set $T_{\mathcal{D}_r}$ according to a probability distribution from the family of distributions $\mathcal{D}_r$. The instance set $T_{\mathcal{D}_r}$ contains all training examples corresponding to strings up to length L, where L is an integer, $L \geq 2n$. It is assumed that the set of all strings up to length $L-1$ with at most one symbol change (i.e. $\mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}$) contains a *teaching set*. For the purposes of the learning algorithm, a teaching set is a set of n strings with linearly independent state distribution vectors. Note that for HMMs satisfying the conditions specified in Sec. 2.2.4, Theorem 2.4 guarantees the presence of a teaching set among the strings in $\mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}$.

Without loss of generality, for the remainder of the chapter, it will be assumed that $n \geq 2$, and $L = 2n \geq 4$.

It is shown next that the unsupervised learning algorithm *PACLearnHMM* of Fig. 4.2, is a PAC-learning algorithm under the helpful distributions $\mathcal{D}_r$.

Theorem 4.2. *Let* $U = (Q, \Sigma, \vec{M}, \left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}, \vec{p})$ *be a target* HMM. *Let* $L > 2n$ *be an integer. Let*

$$T_{\mathcal{D}} = \left\{ \left\langle\, x \,, \vec{P}_U(x) \,\right\rangle : x \in \Sigma^+ \,, |x| \leq L \right\}$$

be an instance set of examples. Let EX be an oracle that draws examples from $T_{\mathcal{D}}$ *according to a probability distribution from the class of distributions* $\mathcal{D}_r$, $r \geq 2$ *defined in* (4.3). *If* $T_{\mathcal{D}}$ *contains a teaching set* $T_S = \{\langle\, x_1 \,, \vec{v}_1 \,\rangle \,, \langle\, x_2 \,, \vec{v}_2 \,\rangle \,, \ldots \,, \langle\, x_n \,, \vec{v}_n \,\rangle\}$ *such that* $\{\vec{v}_1, \vec{v}_2 \ldots, \vec{v}_n\}$ *is a linearly independent set and* $X = \{x_1, x_2, \ldots, x_n\} \subset \mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}$ *then* U *is PAC-learnable under the family of helpful distributions* $\mathcal{D}_r$.

Proof. Consider the set of strings of length up to $L-1$ containing at most one symbol change, namely the set $\mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}$.

It is not difficult to see that given the fact that $X \subset \mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}$, the set of all suffixes to the strings in X, namely X', is contained in the set $\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$, since $\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1}$ contains all the suffixes to strings in the set $\mathcal{X}_{L-1,0}$ and the set $\mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$ contains all suffixes to the strings in the set $\mathcal{X}_{L-1,1}$. Note as well that $\Sigma = \mathcal{X}_{1,0}$.

Therefore:

$$\Sigma \cup X \cup X' \subset \mathcal{X}_{1,0} \cup (\mathcal{X}_{L-1,0} \cup \mathcal{X}_{L-1,1}) \cup (\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}) \quad .$$

However, since $\mathcal{X}_{1,0} \subset \mathcal{X}_{L-1,0} \subset \mathcal{X}_{L,0}$ and $\mathcal{X}_{L-1,1} \subset \mathcal{X}_{L,1}$:

$$\boxed{\Sigma \cup X \cup X' \subset \mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}} \quad .$$

It will be shown that the set of examples corresponding to the strings in the set $\Sigma \cup X \cup X'$ can be obtained from a training set of polynomial size by showing that the sample complexity of the superset given by $\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$ is polynomial in ϵ^{-1}, θ^{-1} , m, and n .

Let $\Upsilon(h, N)$ denote a function such that for any real number $h > 1$, and any positive integer N, $\Upsilon(h, N)$ is the smallest number of independent Bernoulli trials each with probability at least h^{-1} of success, after which the probability of having fewer than N successes is less than h^{-1}. The function $\Upsilon(h, N)$ is known to be bounded (Valiant 1984) by:

$$\Upsilon(h, N) \leq 2h \times (N + \ln(h)) \quad .$$

A success for the purposes of the learning algorithm, is defined as the drawing of a new example pair $\left\langle x \, , \vec{P}_U(x) \right\rangle$ such that $x \in \mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$.

Let $\mathcal{D}_{min_r} = \min\{\mathcal{D}_r(x) : x \in \mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}\}$.

Let $h = \max\left\{\theta^{-1}, \epsilon^{-1}, \mathcal{D}_{min_r}^{-1}\right\}$. Then $h^{-1} \leq \min\left\{\theta, \epsilon, \mathcal{D}_{min_r}\right\}$.

Let $N = |\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}|$. Then:

$$\begin{aligned} N &= \Phi(L,0) + \Phi(L,1) + \Phi(L,2) \\ &= m \times (m-1)^0 \times \binom{L}{0+1} + m \times (m-1)^1 \times \binom{L}{1+1} + m \times (m-1)^2 \times \binom{L}{2+1} \end{aligned}$$

$$= m \times L + m \times (m-1) \times \frac{L \times (L-1)}{2} + m \times (m-1)^2 \times \frac{L \times (L-1) \times (L-2)}{6}$$
$$= (m \times L) \times \left(1 + \frac{(m-1) \times (L-1)}{2} + \frac{(m-1)^2 \times (L-1) \times (L-2)}{6}\right)$$
$$= (m \times L) \times \left(1 + \frac{(m-1) \times (L-1)}{2} \times \left(1 + \frac{(m-1) \times (L-2)}{3}\right)\right) \tag{4.4}$$

From (4.4) it can be seen that $N \approx (m \times L)^3 = (m \times 2n)^3 = \mathcal{O}(m^3 n^3)$. Hence $2h \times (N + \ln(h))$ is polynomial in $\epsilon^{-1}, \theta^{-1}, m$, and n , and bounds $\Upsilon(h, N)$.

After the learning algorithm has gathered $\Upsilon(h, N)$ training examples, either it has seen all strings from $\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$, or it has a probability at least $\mathcal{D}_r(x)$ of drawing a missing string x from the instance set. The latter case implies that after performing $\Upsilon(h, N)$ independent Bernoulli trials each with probability of success greater or equal than h^{-1}, there have not been yet N successes, and the probability of this happening by the definition of $\Upsilon(h, N)$, is less than $h^{-1} \leq \theta$. □

4.4 PAC Learning Algorithm

Figure 4.2 shows a PAC-learning algorithm for the parameters of an HMM under the class of helpful distributions $\mathcal{D}_r$. Algorithm *PACLearnHMM* takes as input parameters the state set Q and alphabet Σ of the target HMM, the requested precision ϵ, and confidence θ, the maximum length L of the strings in the instance set $T_{\mathcal{D}}$, and the parameter r identifying the helpful distribution $\mathcal{D}_r$.

The algorithm attempts to draw all $N = \Phi(\, L,\, 0\,) + \Phi(\, L,\, 1\,) + \Phi(\, L,\, 2\,)$ examples in the set $\mathcal{X}_{L,0} \cup \mathcal{X}_{L,1} \cup \mathcal{X}_{L,2}$, by requesting a training set of size $S = 2 \times h \times (\, N + \ln(h)\,)$ from the oracle EX (lines 1–5).

The value of h is computed as the maximum of $\epsilon^{-1}, \theta^{-1}$, and $\mathcal{D}_{min_r}$ (line 3). Since $\mathcal{D}_r(x)$ only depends on the number of symbol changes $\Psi(x)$ in string x, line 2 of the algorithm computes $\mathcal{D}_{min_r}$ as MIN$\{\, \mathcal{D}(r, 0),\, \mathcal{D}(r, 1),\, \mathcal{D}(r, 2)\,\}$ where $\mathcal{D}(r, c) = \mathcal{D}_r(x)$ such

Algorithm *PACLearnHMM*$(Q, \Sigma, \epsilon, \theta, L, r)$

1. $N \longleftarrow \Phi(\,L,\,0\,) + \Phi(\,L,\,1\,) + \Phi(\,L,\,2\,)$;
2. $\mathcal{D}_{min_r} \longleftarrow \text{MIN}\{\,\mathcal{D}(\,r,\,0\,),\,\mathcal{D}(\,r,\,1\,),\,\mathcal{D}(\,r,\,2\,)\,\}$;
3. $h \longleftarrow \text{MAX}\{\,\epsilon^{-1},\,\theta^{-1},\,\mathcal{D}_{min_r}^{-1}\,\}$;
4. $S \longleftarrow 2 \times h \times (\,N + \ln(h)\,)$;
5. $TrainingSet \longleftarrow \text{EX}(\,S\,)$;
6. $\vec{B} \longleftarrow \text{EMPTY(Matrix)}$; $\vec{p} \longleftarrow \vec{0}_n$;
7. for each $< x\,,\,\vec{d}_x > \in TrainingSet$ do
8. if $(x \in \Sigma)$ then
9. $\vec{p} \longleftarrow \vec{p} + \vec{d}_x$;
10. end ;
11. if $(|x| < L)$ and $(\vec{d}_x \notin \text{SPAN}(\vec{B}))$ then
12. if (FIND_SUFFIXES($TrainingSet$, x, Σ)) then
13. $\vec{B} \longleftarrow \text{APPEND\_ROW}(\,\vec{d}_x\,)$;
14. end ;
15. end ;
16. end ;
17. if $(\sum_{i=1}^{|Q|} \vec{p}[i] \neq 1)$ return (not found) ;
18. for each $\sigma \in \Sigma$ do
19. $\vec{W}_\sigma \longleftarrow \text{EMPTY(Matrix)}$;
20. for each $\vec{d}_x \in \vec{B}$ do
21. $< x\sigma\,,\,\vec{d}_{x\sigma} > \longleftarrow \text{GET\_EXAMPLE}(TrainingSet,\,x\sigma\,)$;
22. $\vec{W}_\sigma \longleftarrow \text{APPEND\_ROW}(\,\vec{d}_{x\sigma}\,)$;
23. end ;
24. end ;
25. $\vec{W} \longleftarrow \sum_{\sigma \in \Sigma} \vec{W}_\sigma$;
26. solve for $\vec{M}$ the matrix system:

$$\begin{array}{rcl} \vec{B} \cdot \vec{M} & = & \vec{W} \\ \vec{M} \cdot \vec{1}_n^{\,T} & = & \vec{1}_n^{\,T} \\ \vec{M} & \geq & \vec{0}_{n \times n} \end{array} ;$$

27. solve for the matrices D_σ the following system of matrix equations:

$$\left.\begin{array}{rcl} \vec{B} \cdot \vec{M} \cdot \vec{D}_\sigma & = & \vec{W}_\sigma \\ \vec{D}_\sigma & \geq & \vec{0}_{n \times n} \end{array}\right\} \forall (\sigma \in \Sigma)$$
$$\begin{array}{rcl} \sum_{\sigma \in \Sigma} \vec{D}_\sigma & = & \vec{I}_n \end{array} ;$$

28. if solutions were found for $\vec{M}$, and each $\vec{D}_\sigma$ then:
29. return $(Q, \Sigma, \vec{M}, \left\{\vec{D}_{\sigma_1}, \ldots, \vec{D}_{\sigma_m}\right\}, \vec{p})$;
30. else return (not found) ;

Figure 4.2: Algorithm PACLearnHMM to learn the parameters of an HMM.

that $\Psi(x) = c$:

$$\mathcal{D}(r,c) = \frac{(r-1)\ r^{(L-c-1)}}{m(m-1)^c \binom{L}{c+1} (r^L - 1)} \ .$$

Each example $\left\langle x\,, \vec{P}_U(x) \right\rangle$ from the drawn training set of examples is examined. The state distribution vectors for examples with strings from Σ (strings of length one) are summed up to obtain the ISPD (lines 8–10). If examples corresponding to all m symbols in the alphabet Σ are not contained in the training set, the algorithm terminates (line 17), since those examples are required to compute the exact HMM ISPD.

While examining the training set, only those state distribution vectors for strings of length less than L that increase the rank of the matrix $\vec{B}$ and for which all one-symbol string suffixes are present in the training set, are appended to the matrix $\vec{B}$ (lines 11–15). Line 12 calls the function FIND_SUFFIXES to search the training set looking for the presence of all one-symbol suffixes of a string. The function returns 'true' if all the suffixes are found in the training set, or 'false' otherwise.

In lines 18–24, the algorithm constructs the matrices of states distribution vectors corresponding to all the one-symbol suffixes of the strings whose state distribution vectors are the rows of $\vec{B}$. The suffix state distribution vectors are obtained by calling the function GET_EXAMPLE (line 21) which searches the training set. The suffix examples are guaranteed to be found by the function, since only strings for which all suffixes are in the training set where considered to be part of the basis $\vec{B}$.

Lines 25–27 proceed to solve the TPD and DPD matrix systems in identical manner as explained for the supervised learning algorithm. The time-complexity of the algorithm is polynomial in the number of examples S in the training set. Worse case scenario, if each of the S training examples were candidates to be appended to the basis $\vec{B}$ (which is a considerable overestimation given that only strings of length up to $L-1$ are considered), the function FIND_SUFFIXES would have to search the training set looking for each

of the m suffixes of every one of the S strings in the training set. Since in the worst case, each of the S examples would have to be examined for each of the m suffixes, $S \times m \times S = m \times S^2$ string comparisons would take place. Since S, the training set size, is a polynomial function of ϵ^{-1}, θ^{-1}, m, and n, if follows that $m \times S^2$ is polynomial as well.

Similarly, getting all m suffixes from the training set for each of the n strings whose state distribution vectors form the rows of $\vec{B}$ in lines 18—24 involves searching the training set $n \times m$ times, requiring a worse case number of $n \times n \times S$ string comparisons.

Chapter 5:
Hybrid HMM Parameter Approximation From State Distribution Vectors

Most commonly used techniques to attain HMM parameter approximations, such as the Baum-Welch algorithm, train the model by using only positive examples — sequences that belong to the model and are assumed to have high generating probability. In contrast, both algorithms presented so far, *SupLearnHMM* and *PACLearnHMM*, attempt to learn the parameters of an HMM exactly —rather than an approximation— from strings representing a mixture of both positive and negative examples —strings with high and low generating probability in the model, respectively.

An interesting hybrid approach to HMM parameter approximation from both positive and negative examples, the MA algorithm (Mamitsuka 1997), utilizes a training set of strings labeled with their generating probabilities. The MA algorithm iterates from an initial parameter guess attempting to minimize a type of error distance between the current and the target model generating probabilities of each string in the training set. In this chapter the updating rules of the MA algorithm will be described and a new modified version, The MASD algorithm will be introduced. The MASD algorithm trains HMM parameters from a set of strings and their corresponding state distribution vectors. By incorporating additional information about the strings in the training set (their state distribution vectors) the algorithm attains, in the same number of iterations, significantly closer approximations to the target HMM parameters than the original MA algorithm. Experimental data from simulation runs for both algorithms supporting this claim will be presented.

5.1 The MA Algorithm

The MA algorithm is a smooth algorithm for sequence discrimination using a dataset of examples consisting of strings and their target likelihoods. The HMM model assumed by the algorithm MA differs slightly from the definitions given before in the sense that it assumes that there is a state in the model from which each all transitions start and that does not emit any display symbols. This discrepancy is easy to overcome, by first transforming an HMM $U = (Q, \Sigma, \delta, \beta, \rho)$ into an equivalent HMM $U^{\#} = (Q^{\#}, \Sigma, \delta^{\#}, \beta, \rho^{\#})$ that incorporates an additional state q_0 which does not display any symbol, and hence does not have an associated display distribution:

- $Q^{\#} = Q \cup \{q_0\}$, where q_0 is the initial state in $U^{\#}$,
- $\delta^{\#}$ is defined as:

$$\begin{aligned} \delta^{\#}(q_i, q_j) &= \delta(q_i, q_j), && \forall (1 \leq i, j \leq |Q|) \\ \delta^{\#}(q_0, q_j) &= \rho(q_j), && \forall (1 \leq j \leq |Q|) \\ \delta^{\#}(q_i, q_0) &= 0, && \forall (0 \leq i \leq |Q|)\,. \end{aligned}$$

- $\rho^{\#}$ is defined as:

$$\rho^{\#}(q_i) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall (0 \leq i \leq |Q|)\,.$$

The scheme associates the ISPD of the HMM U with the transitions from the initial state q_0 to every other state in $U^{\#}$.

In order to describe the MA algorithm several definitions will be introduced first.

Definition 5.1. Let $x = o_1 o_2 \cdots o_k \in \Sigma^+$ be a string of length $|x| = k$. For each $(1 \leq t \leq k)$, and $(0 \leq j \leq |Q|)$, The *forward probability* $\alpha_t(j)$ is the probability that the partial string $o_1 \cdots o_t$ is generated and that the state at time t is q_j:

$$\alpha_t(j) = Pr(o_1 o_2 \cdots o_t\,,\, s_t = q_j \,|\, U^{\#})\ .$$

Note that $\alpha_t(j)$ can be computed using the forward algorithm since $\alpha_t(j) = \vec{P}_{U^\#}[j]$.

For all $0 \leq j \leq |Q| = n$:

$$\alpha_0(j) = \begin{cases} 1 & \text{if } j = 0\,, \\ 0 & \text{otherwise}\,. \end{cases} \quad ,$$

$$\alpha_t(j) = \sum_{i=1}^{n} \delta(q_i, q_j)\, \beta(q_j, o_t)\, \alpha_{t-1}(i)\,, \qquad \forall (1 \leq t \leq k) \ \ ,$$

$$\alpha_{k+1}(j) = \sum_{i=1}^{n} \delta(q_i, q_j)\, \alpha_k(i) \ \ .$$

In a similar manner, the backward probability $\varphi_t(i)$ can be defined:

Definition 5.2. Let $x = o_1 o_2 \cdots o_k \in \Sigma^+$ be a string of length $|x| = k$. For each $(1 \leq t \leq k)$, and $(0 \leq j \leq |Q|)$, The *backward probability* $\varphi_t(i)$ is the probability that the partial string $o_{t+1} \cdots o_k$ is generated and that the state at time t is q_i:

$$\varphi_t(i) = Pr(o_{t+1} o_{t+2} \cdots o_k \,|\, s_t = q_i\,, U^\#) \ \ .$$

The backward probability $\varphi_t(i)$ can be computed recursively:

For all $0 \leq i \leq |Q| = n$:

$$\varphi_{k+1}(i) = \begin{cases} 0 & \text{if } i = 0\,, \\ 1 & \text{otherwise}\,. \end{cases} \quad ,$$

$$\varphi_k(i) = \sum_{j=1}^{n} \delta^\#(q_i, q_j)\, \varphi_{k+1}(j) \ \ ,$$

$$\varphi_t(i) = \sum_{j=1}^{n} \delta^\#(q_i, q_j)\, \beta^\#(q_j, o_{t+1})\, \varphi_{t+1}(j) \qquad \forall (0 \leq t < k) \ \ .$$

Note that the generating probability of the string x is:

$$Pr(x \,|\, U^\#) = \sum_{i=0}^{n} \alpha_0(i)\, \varphi_0(i) = \sum_{i=0}^{n} \alpha_{k+1}(i)\, \varphi_{k+1}(i) = \sum_{i=1}^{n} \alpha_k(i) \ \ .$$

Now consider the probability $\gamma_t(i) = Pr(s_t = q_i \,|\, x, U^{\#})$, the probability of being in state q_i at time t given the string x and the model. $\gamma_t(i)$ can be expressed in terms of the forward and backward probabilities:

$$\gamma_t(i) = \frac{\alpha_t(i)\,\varphi_t(i)}{Pr(x \,|\, U^{\#})} \ .$$

Finally, the last variable needed to describe the MA algorithm is $\xi_t(i,j)$, the probability of being in state q_i at time t and state j at time $t+1$ given the string x and the model:

$$\xi_t(i,j) = Pr(s_t = q_i, s_{t+1} = q_j \,|\, x,\, U^{\#})$$

$\xi_t(i,j)$ can be computed as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i)\,\delta^{\#}(q_i,q_j)\,\beta^{\#}(q_j,o_{t+1})\,\varphi_{t+1}(j)}{Pr(x \,|\, U^{\#})} \qquad \forall (0 \le t < k)\,,$$
$$\xi_k(i,j) = \frac{\alpha_k(i)\,\delta^{\#}(q_i,q_j)\,\varphi_{k+1}(j)}{Pr(x \,|\, U^{\#})} \ .$$

Let T be a training set of pairs $\langle\, x\,, \hat{p}_x \,\rangle$, where x is a string in the model and $\hat{p}_x = Pr(x \,|\, U)$ is its target generating probability. Let $p_x = Pr(x \,|\, U^{\#})$ be the generating probability of x in the current model $U^{\#}$.

Let

$$\Delta_x = \log(\frac{\hat{p}_x}{p_x}) \quad \text{and} \quad \Delta_{max} = \log(\frac{\hat{p}_{max}}{\hat{p}_{min}}) \ ,$$

where $\hat{p}_{max} = \max\{\, \hat{p}_x : \langle\, x\,, \hat{p}_x \,\rangle \in T \,\}$ and $\hat{p}_{min} = \min\{\, \hat{p}_x : \langle\, x\,, \hat{p}_x \,\rangle \in T \,\}$

The algorithm begins by constructing an initial guess for $\delta^{\#}$ and $\beta^{\#}$ computed by using the following Boltzman distribution equations:

$$\delta^{\#}(q_i,q_j) = \frac{e^{\lambda w_{i,j}}}{\sum_{l=1}^{n} e^{\lambda w_{i,l}}} \ , \qquad \beta^{\#}(q_j,\sigma) = \frac{e^{\lambda v_{j,\sigma}}}{\sum_{o\in\Sigma}^{n} e^{\lambda v_{j,o}}} \ , \tag{5.1}$$

where λ is a constant.

Initially, the variables v and w are allowed to take any real positive values satisfying $\sum_{j=1}^{n} w_{i,j} = 1$, and $\sum_{\sigma \in \Sigma} v_{j,\sigma} = 1$. In each iteration, the algorithm updates the values of w and v to minimize the function :

$$\sum_{\langle\, x\,,\hat{p}_x\,\rangle} -\log\left(\frac{\Delta_{max}^2 - \Delta_x^2}{\Delta_{max}^2}\right)$$

with the goal of making Δ_x approximate the value zero for each string in the training set. The smooth updating rules used by the algorithm are shown below:

$$w_{i,j}^{(new)} = w_{i,j}^{(old)} + C_a \sum_{\langle\, x\,,\hat{p}_x\,\rangle} \left(\frac{\Delta_x}{(\Delta_{max}^2 - \Delta_x^2)} \sum_{t=1}^{k} \left(\xi_t(i,j) - \delta^{\#}(q_i, q_j)\,\gamma_t(i)\right)\right) \ ,$$

$$v_{j,\sigma}^{(new)} = v_{j,\sigma}^{(old)} + C_b \sum_{\langle\, x\,,\hat{p}_x\,\rangle} \left(\frac{\Delta_x}{(\Delta_{max}^2 - \Delta_x^2)} \sum_{t=1}^{k} \left(\gamma_t(j)_{o_t=\sigma} - \beta^{\#}(q_j, \sigma)\,\gamma_t(j)\right)\right) \ ,$$

where C_a and C_b are constants.

At the end of each iteration, new values for $\delta^{\#}$ and $\beta^{\#}$ are computed from the updated variables w, and v by means of (5.1).

5.2 The MASD Algorithm

The MASD algorithm proposed here, is a modified version of the MA algorithm to approximate the parameters of an HMM from a training set T consisting of pairs $\left\langle\, x\,,\vec{P}_U(x)\,\right\rangle$ of strings and their corresponding state distribution vectors. The use of state distribution vectors improves the approximation obtained from the training process as shown in the next section.

New notation is introduced next in order to describe the algorithm:
Let $\vec{P}_{max}$, $\vec{P}_{min}$, and $\vec{\Delta}_{max}$ be row vectors such that for $i = 1, \ldots, n$:

$$\vec{P}_{max}[i] = \max\{\, \vec{P}_U(x)[i] : \left\langle\, x\,,\vec{P}_U(x)\,\right\rangle \in T\,\} \ ,$$

$$\vec{P}_{min}[i] = \min\{\, \vec{P}_U(x)[i] : \left\langle\, x\,, \vec{P}_U(x)\, \right\rangle \in T\,\} \ , \text{ and}$$

$$\vec{\Delta}_{max}[i] = \log\left(\frac{\vec{P}_{max}[i]}{\vec{P}_{min}[i]}\right) \ .$$

Let $\vec{\Delta}_x$ be row vector for each string x in the training set consisting of the log difference between the forward vectors $\vec{P}_U(x)$ in the target model and in the current approximation $\vec{P}_{U^\#}(x)$:

$$\vec{\Delta}_x[i] = \log\left(\frac{\vec{P}_U(x)[i]}{\vec{P}_{U^\#}(x)[i]}\right) \ , \qquad \forall(1 \le i \le n) \ .$$

The updating rules for the MASD algorithm are:

$$w_{i,j}^{(new)} = w_{i,j}^{(old)} + C_a \sum_{\langle\, x\,, \hat{p}_x\,\rangle} \left(\left(\sum_{l=1}^{n} \frac{\vec{\Delta}_x[l]}{(\vec{\Delta}^2_{max}[l] - \vec{\Delta}^2_x[l])} \right) \sum_{t=1}^{k} \left(\xi_t(i,j) - \delta^\#(q_i, q_j)\, \gamma_t(i) \right) \right) \ ,$$

$$v_{j,\sigma}^{(new)} = v_{j,\sigma}^{(old)} + C_b \sum_{\langle\, x\,, \hat{p}_x\,\rangle} \left(\left(\sum_{l=1}^{n} \frac{\vec{\Delta}_x[l]}{(\vec{\Delta}^2_{max}[l] - \vec{\Delta}^2_x[l])} \right) \sum_{t=1}^{k} \left(\gamma_t(j)_{o_t=\sigma} - \beta^\#(q_j, \sigma)\, \gamma_t(j) \right) \right) \ ,$$

where C_a and C_b are constants.

As in the original algorithm MA, at the end of each iteration the new values obtained for w, and v are used to compute new estimations $\delta^\#$ and $\beta^\#$ from (5.1).

5.3 Comparative Simulation Results

With the goal of comparing the performance of algorithms MA and MASD, the following experiment was conducted:

- Nine (9) target HMMs where randomly generated each with an alphabet of five display symbols and different number of states $n \in \{2, 5, 6, 7, 8, 10, 15, 20, 25\}$
- A dataset of sixty (60) strings of varied lengths was generated for each target HMM, and randomly divided into a training dataset of forty (40) strings and a testing set of twenty (20) strings. The same datasets were used to train and test both algorithms.

– For each target HMM five (5) runs of each algorithm were performed each consisting of 50 iterations on the same dataset but differing on the initial guess parameters utilized, which were randomly generated for each of the five runs.

– Every run of each algorithm used the values $\lambda = C_a = C_b = 1$ for the smoothing and updating steps.

Each algorithm's performance was measured by computing the average sum of squared error (SSE) and average Kullback-Leibler divergence (KLD) incurred by the approximations on the strings of the testing dataset. Let V be the set of strings in the testing set, then:

– the average sum of squared error was calculated for each algorithm as:

$$SSE(V) = \frac{1}{|V|} \sum_{x \in V} \left(\hat{p}_x - p_x\right)^2 \ ,$$

– the average Kullback-Leibler divergence of the approximation with respect of the target HMM was computed as:

$$KLD(V) = \frac{1}{|V|} \sum_{x \in V} \hat{p}_x \log_2 \left(\frac{\hat{p}_x}{p_x}\right) \ .$$

Table 5.1 summarizes the results of the experiment. For each target HMM the data corresponding to the run with the smallest SSE error is shown (the one with minimum SSE error for either the MA or the MASD algorithm runs). All runs were performed with a random training dataset of forty (40) strings and testing set of twenty (20) strings. The number of states and maximum string length in the dataset for each run is shown on the table. Both algorithms performed fifty (50) iterations in each run. The average sum of squared error and average Kullback-Leibler divergence of each run are shown for each algorithm. The table shows algorithm MASD's approximation to be a significant

Table 5.1: Simulation results for several runs of algorithms MA and MASD.

Number of States	Longest String	Algorithm MA		Algorithm MASD	
		SSE	KLD	SSE	KLD
2	10	0.03950	2.190×10^{-4}	0.02962	1.987×10^{-4}
5	15	0.04150	-2.816×10^{-3}	0.04717	-2.702×10^{-3}
6	20	0.08074	2.556×10^{-4}	0.04616	-2.025×10^{-4}
7	20	0.09241	4.643×10^{-3}	0.03983	3.801×10^{-3}
8	20	0.08034	1.135×10^{-3}	0.02229	-3.049×10^{-4}
10	20	0.09649	2.964×10^{-4}	0.07764	2.737×10^{-4}
15	20	0.08314	1.766×10^{-4}	0.03189	1.923×10^{-4}
20	25	0.09131	1.250×10^{-4}	0.01923	-1.846×10^{-4}
25	30	0.12016	2.829×10^{-3}	0.03132	1.115×10^{-3}

improvement over the HMM estimations obtained by algorithm MA.

Figures 5.1 and 5.2 show a plot of the SSE and KLD for each algorithm as a function of the number of states in the target HMM.

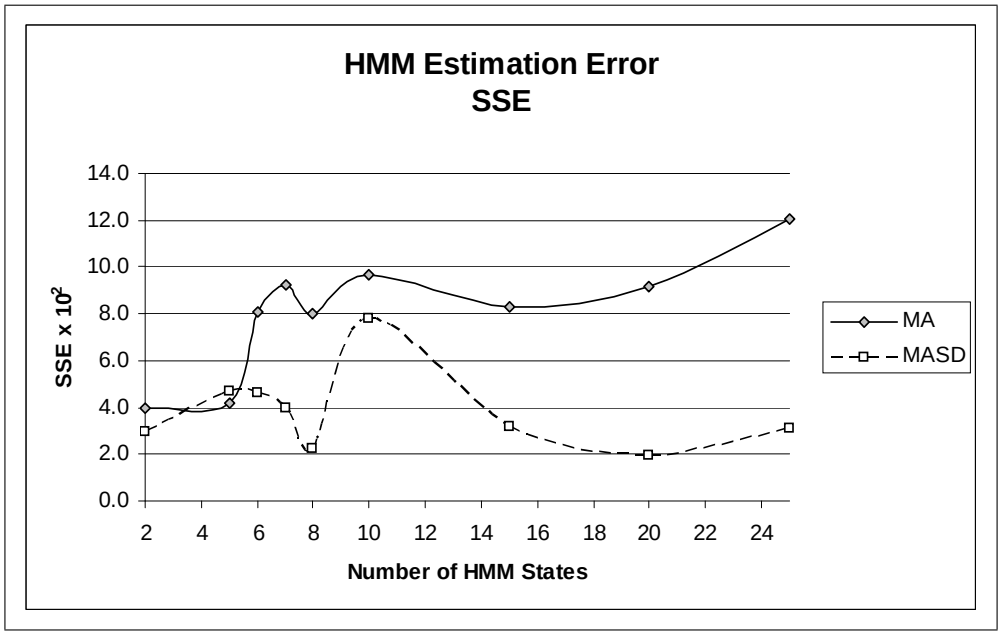

Figure 5.1: Average sum of squared error ($\times100$) of algorithms MA and MASD.

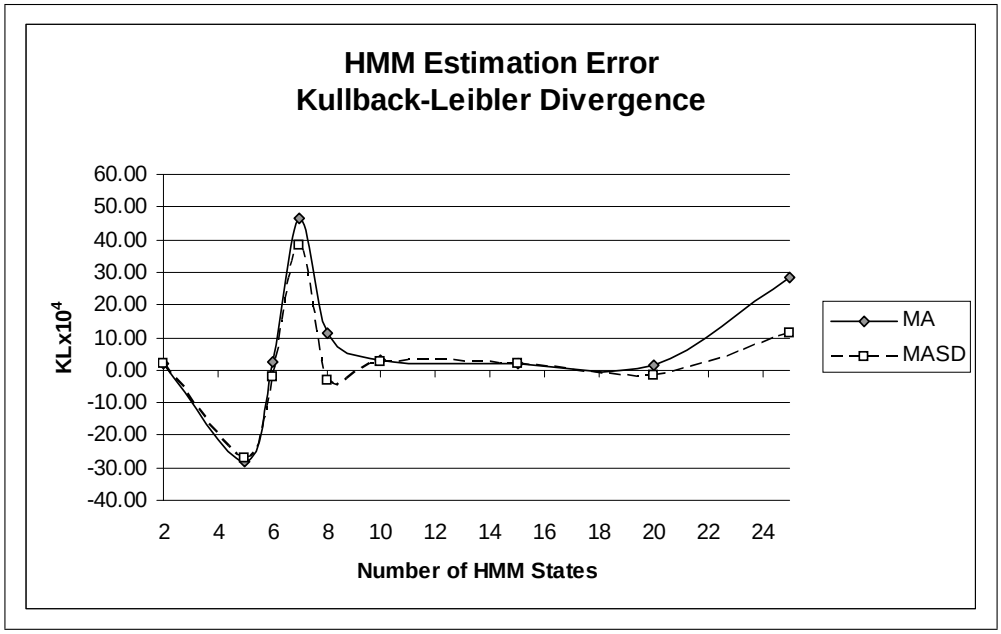

Figure 5.2: Average Kullback-Leibler divergence ($\times 10^4$) of algorithms MA and MASD.

Chapter 6:
Conclusions and Future Directions

In this research, we have proposed a polynomial-time supervised learning algorithm for training the parameters of a discrete first-order hidden Markov model using state distribution vectors provided by the SD oracle. The algorithm departs from other current approaches in that it attempts to learn the target HMM parameters exactly, rather than approximating the model by maximizing the likelihood of positive sequence observations. We justified the use of the SD oracle, by proving a theorem stating the NP-Completeness of the consistency problem for hidden Markov models using state distribution vectors.

We have proved the correctness of the learning algorithm and demonstrated its complexity to be a polynomial function of the size of its input. We describe the conditions under which the algorithm is guaranteed to obtain full basis from the state distribution vectors provided by the SD oracle. In order to confirm the viability of the approach, we conducted a simulation run of the algorithm on randomly generated HMMs.

In the setting of unsupervised learning, we have elaborated an alternative learning framework for efficient HMM learning in the absence of the SD oracle. We defined a class of helpful probability distributions and proved that HMMs can be learned in polynomial-time in the PAC setting under these helpful distributions. Moreover, we proposed an efficient PAC-learning algorithm for learning HMM parameters under our family of helpful distributions.

Additionally, we have proposed a hybrid approach to learning HMMs from state distribution vectors that computes an approximation of the HMM parameters from a set of strings and their respective state distribution vectors. We have presented experimental results showing the approach to be an improvement over a version of the algorithm that trains the HMM parameters from the string generating probabilities.

With respect to future directions, one line of work is to obtain a set of more general conditions for the existence of a basis of state distribution vectors. Although certainly sufficient, the conditions given are not necessary for a basis to be found. Moreover, our simulation results show that a basis could be found in over 99% of the randomly generated HMMs by performing a linear (usually just n) number of state distribution queries, which suggest that it may be possible to establish more general conditions for their existence.

Further research would also be beneficial on alternative ways to obtain state distribution vectors such as combining the information from several experts. We have shown that state distribution vectors are powerful tools for exact and approximate learning of HMM parameters. However, obtaining such information could prove to be difficult, since for most applications state distribution vectors are not readily available. It may be possible, however, to obtain the state distribution vectors from the knowledge of two or more experts. For example, since $P_U(x)[i] = Pr(\, x,\, s_t = q_i \,|\, U)$ from two oracles, one providing relative state frequencies for a string x, $Pr(s_t = q_i \,|\, x, U)$ and another supplying the string generating probability $Pr(\, x \,|\, U)$, the state distribution vector for x can be obtain as $P_U(x)[i] = Pr(s_t = q_i \,|\, x, U) \times Pr(\, x \,|\, U)$.

In summary we have proposed, as our main contribution, two algorithms for supervised and unsupervised learning of HMM parameters from state distribution vectors as well as a hybrid algorithm to approximate HMM parameters. Additionally, we proved the important theoretical result stating the NP-Completeness of the consistency problem for HMMs using state distribution vectors.

References

Abe, N. and Warmuth, M. K.: 1992, On the computational complexity of approximating distributions by probabilistic automata, *Machine Learning* **9**(2-3), 205–260.

Angluin, D.: 1987, Learning regular sets from queries and counterexamples, *Information and Computation* **75**, 87–106.

Angluin, D.: 1988, Queries and concept learning, *Machine Learning* **2**(4), 319–342.

Baggenstoss, P. M.: 2001, A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces, *IEEE Transactions on Speech and Audio Processing* **9**(4), 411–416.

Baldi, P. and Chauvin, Y.: 1994, Smooth on-line learning algorithms for hidden Markov models, *Neural Computation* **6**(2), 307–318.

Baldi, P., Chauvin, Y., Hunkapiller, T. and McClure, M. A.: 1993, Hidden Markov models in molecular biology: New algorithms and applications, *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, Morgan Kaufmann Publishers Inc., pp. 747–754.

Baum, L. E.: 1972, An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes, *Inequalities* **2**(1–8).

Baum, L. E. and Petrie, T.: 1966, Statistical inference for probabilistic functions of finite state Markov chains, *Annals of Mathematical Statistics* **37**(1554–1563).

Baum, L. E., Petrie, T., Soules, G. and Weiss, N.: 1971, A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains, *Annals of Mathematical Statistics* **41**(1).

Berman, A. and Plemmons, R. J.: 1979, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, New York.

Bshouty, N., Cleve, R., Gavaldà, R., Kannan, S. and Tamon, C.: 1996, Oracles and queries that are sufficient for exact learning, *Journal of Computer and System Sciences* **52**(3), 421–433.

Bshouty, N. H., Jackson, J. C. and Tamon, C.: 2002, Exploring learnability between exact and PAC, *Proceedings of the 15th Annual Conference on Computational Learning Theory*, Springer-Verlag, pp. 244–254.

Chan, T. F.: 1982, An improved algorithm for computing the singular value decomposition, *ACM Transactions on Mathematical Software* **8**(1), 72–83.

Davis, R. I. A., Lovell, B. C. and Caelli, T.: 2002, Improved estimation of hidden Markov model parameters from multiple observation sequences, *16th International Conference on Pattern Recognition: ICPR'2000*, Vol. 2, IEEE Press, pp. 168–171.

Dempster, A. P., Laird, N. M. and Rubin, D. B.: 1977, Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)* **39**(1), 1–38.

Denis, F. and Gilleron, R.: 1997, PAC learning under helpful distributions, *Proceedings of the 8th International Conference on Algorithmic Learning Theory*, Springer-Verlag, pp. 132–145.

Durbin, R., Eddy, S., Krogh, A. and Mitchison, G.: 1998, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge University Press, Cambridge, United Kingdom.

Eddy, S.: 1996, Hidden Markov models, *Current Opinion in Structural Biology* **6**(3), 361–365.

Farina, L. and Rinaldi, S.: 2000, *Positive Linear Systems: Theory and Applications*, John Wiley & Sons.

Gold, E. M.: 1978, Complexity of automaton identification from given data, *Information and Control* **37**, 302–320.

Golub, G. H. and Kahan, W.: 1965, Calculating the singular values and pseudo-reverse of a matrix, *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis* **2**(2), 205–224.

Golub, G. H. and Van Loan, C. F.: 1996, *Matrix Computations*, 3rd edn, John Hopkins University Press, Baltimore, Maryland.

Goode, S. W.: 1991, *An Introduction to Differential Equations and Linear Algebra*, Prentice Hall, Englewood Cliffs, New Jersey.

Haussler, D., Krogh, A. and Mian, I. S.: 1994, A hidden Markov model that finds genes in Ecoli DNA, *Nucleic Acids Research* **22**(22), 4768–4778.

Hohn, F. E.: 1958, *Elementary Matrix Algebra*, The MacMillan Company, New York, New York.

Hughey, R. and Krogh, A.: 1996, Hidden Markov models for sequence analysis: extension and analysis of basic method., *Comp. Appl. BioSci* **12**(2), 95–108.

Ito, H., Amari, S.-I. and Kobayashi, K.: 1992, Identifiability of hidden Markov information sources and their minimum degrees of freedom, *IEEE Transactions on Information Theory* **38**(2), 324–333.

Juang, B. and Rabiner, L. R.: 1985, A probabilistic distance measure for hidden Markov models, *AT&T Technical Journal* **64**(2), 391–408.

Juang, B. and Rabiner, L. R.: 1990, The segmental K-means algorithm for estimating parameters of hidden Markov models, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **38**(9), 1639–1641.

Juang, B. and Rabiner, L. R.: 1991, Hidden Markov models for speech recognition, *Technometrics* **33**(3), 251–272.

Karmarkar, N.: 1984, A new polynomial-time algorithm for linear programming, *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, ACM Press, pp. 302–311.

Karplus, K., Sjolander, K. and Sanders, C.: 1997, Predicting protein structure using hidden Markov models, *Proteins* **Supplement**(1), 134–139.

Krogh, A., Brown, M., Mian, I. S., Sjolander, K. and Haussler, D.: 1993, Hidden Markov models in computational biology: Applications to protein modeling, *Technical Report UCSC-CRL-93-32*.

L. Eikvil, R. H.: 2001, Traffic surveillance in real-time using hidden Markov models, *Proceedings of the 12th Scandinavian Conference on Image Analysis (SCIA)*.

Lee, K., Hon, H., Hwang, M. and Huang, X.: 1990, Speech recognition using hidden Markov models: A CMU perspective, *Speech Communication* **9**(5–6), 497–508.

Leroux, B. G.: 1992, Maximum-likelihood estimation for hidden Markov models, *Stochastic Processes and their Applications* **40**, 127–143.

Levinson, S. E., Rabiner, L. R. and Sondhi, M. M.: 1983, An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition, *The Bell System Technical Journal* **62**(4), 1035–1074.

Li, X., Parizeau, M. and Plamondon, R.: 2000, Training hidden Markov models with multiple observations—a combinatorial method, *IEEE Transactions on Patter Analysis and Machine Intelligence* **22**(4), 371–377.

Mamitsuka, H.: 1997, Supervised learning of hidden Markov models for sequence discrimination, *Proceedings of the first annual international conference on Computational molecular biology*, pp. 202–208.

Mitchell, T. M.: 1997, *Machine Learning*, McGraw-Hill.

Qian, W. and Titterington, D. M.: 1991, Estimation of parameters in hidden Markov models, *Phylosophical Transacions: Physical Sciences and Engineering* **337**(1647), 407–428.

Rabiner, L. R.: 1989, A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE* **77**(2), 257–286.

Rabiner, L. R. and Juang, B.: 1986, An introduction to hidden Markov models, *IEEE ASSP Magazine* **3**(1), 4–16.

Scheffer, T., Decomain, C. and Wrobel, S.: 2001, Active hidden Markov models for information extraction, *Lecture Notes in Computer Science* **2189**, 309–318.

Seneta, E.: 1981, *Non-Negative Matrices and Markov Chains*, 2nd edn, Springer Verlag, New York, New York.

Starner, T. and Pentland, A.: 1995, Visual recognition of american sign language using hidden Markov models, *Proceedings of the International Workshop on Automatic Face and Gesture Recognition*, pp. 189–194.

Stolcke, A. and Omohundro, S. M.: 1993, Hidden Markov model induction by bayesian model merging, *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, Morgan Kaufmann Publishers Inc., pp. 11–18.

Tzeng, W.: 1992, Learning probabilistic automata and Markov chains via queries, *Machine Learning* **8**(2), 151–166.

Valiant, L. G.: 1984, A theory of the learnable, *Communication of the ACM* **27**(11), 1134–1142.

Vlontzos, J. A. and Kung, S. Y.: 1992, Hidden Markov models for character recognition, *IEEE Transactions on Image Processing* **1**(4), 539–543.

Vita

Luis Gabriel Moscovich was born in Buenos Aires, Argentina, on February 3, 1968. He attended high school in his natal city of Buenos Aires from 1981 to 1985. In November 1994, he traveled to Baton Rouge, Louisiana, where he began attending Louisiana State University in the Fall of 1995. He completed his bachelor's degree in computer science, *cum laude*, in 1998. In January 1999, he entered the doctoral program in the Department of Computer Science at Louisiana State University and is currently a doctoral candidate under the supervision of Dr. Jianhua Chen. His dissertation defense has been scheduled for December 2004.